\\CSC 478
Spring 2017
Professor Velkoski

Movie Lens Exploration:
Regression, Clustering and a Recommender System

Sarah Cummings, Anika Ghosh, Mitali Ingole

Introduction and Preliminary Data Analysis

The MovieLens data set is an online resource of movie ratings that were collected by GroupLens research and made publically available via https://grouplens.org/datasets/movielens/. For our project, we are using a data set that consists of 100,000 ratings from 943 users on 1682 different movies. The ratings are from 1 to 5, with 1 indicating the movie was unfavorable and 5 indicating the movie was favorable. Also note that each user has rated at least twenty movies. The data was collected during a seven month period from September 19, 1997 through April 22, 1998 via the movie lens website.

The raw data is broken into three files we will focus on: users data, mov\ie data, and rating data. The users data contains the following variables: user id, age, gender, occupation, and zip code. The movies data has the following variables as they pertain to each movie: movie id, movie title, release date, video release data, IMDb url, and nineteen binary columns that correspond to different movie genres. Finally, the ratings file has the user id, item id (which corresponds to the movie id), rating, and timestamp.

For this project, our goal is to gain a better understanding of the users and their ratings. We hope to be able to predict a movie rating for a movie the user has not seen before, and recommend a movie a user would like, We will thus make use of three machine learning techniques: regression, clustering, and recommender systems.

Data Exploration: User’s data

In the users data, we began by examining the distribution of each variable. For the sex variable, we found that there were 670 makes and 273 females. Our data set is highly imbalanced in relation to gender; there are significantly more males than females. For the occupation variable, we found there were 21 occupations accounted for. The most popular occupations were “student”, “other”, and “educator”. The least popular were “doctor”, “homemaker” and “none”. The distribution of these categorical variables can be seen in figures 1 and 2 below. Note that for zip code, also a categorical variable, we had 795 unique zip codes included in the data.

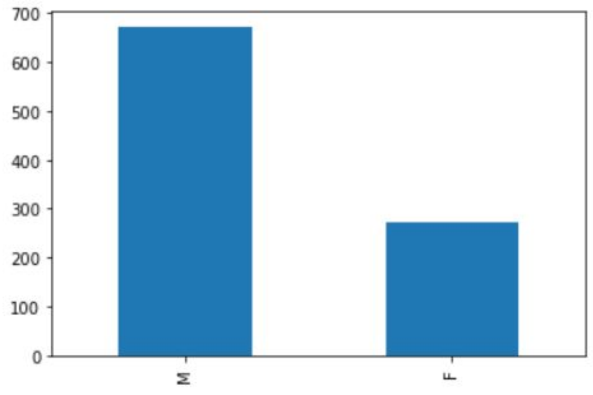


Figure 1: Gender Distribution

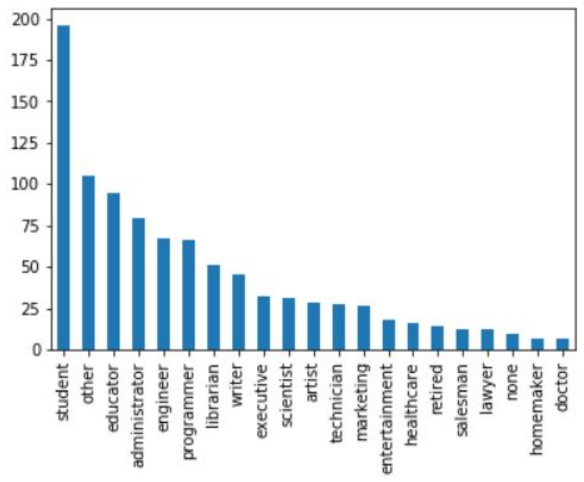


Figure 2: Occupation Distribution

Looking at the age variable, we find that is is skewed left with most users being in their late twenties or early thirties. The minimum age is 7 and the max is 73. The median age is 31, while the mean age is 34. A histogram of the age variable distribution is pictured below in figure 3. We also decided to create equi-width age bins, each spanning 10 years, in case that was useful for our analysis later.

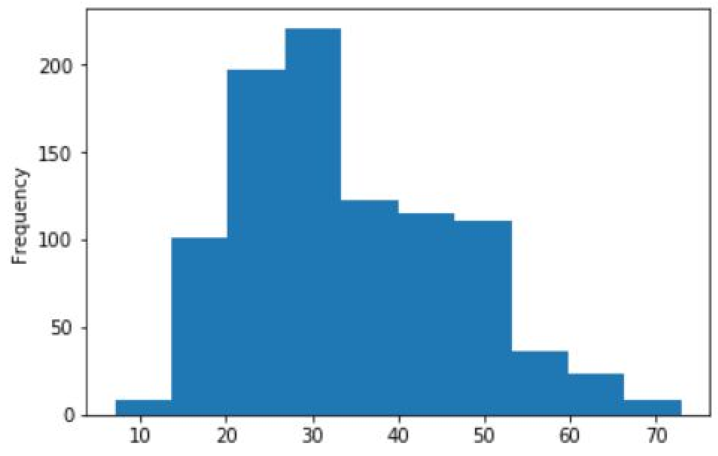
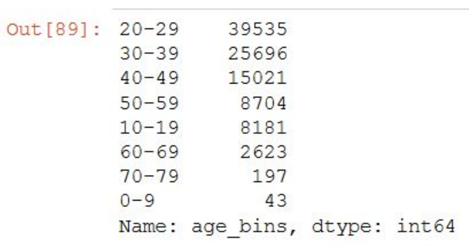


Figure 3: Histogram of Age Variable



```
Out[89]: 20-29    39535
         30-39    25696
         40-49    15021
         50-59     8704
         10-19     8181
         60-69     2623
         70-79      197
         0-9         43
         Name: age_bins, dtype: int64
```

Figure 4: Age Bins Frequency Count

Data Exploration: Movies data and Ratings data

In examining the movies data, we began by seeing how many movies fell into each genre. We found that Drama was the most popular genre, with 725 movies falling into that category. Comedy was the second most popular is comedy with 505 movies, and third is action with 251 movies. Fantasy was the least popular, with 22 movies falling into that genre. A barchart of the genre counts can be seen in Figure 5.

It's important to note that some movies fall into more than one genre. While 833 movies have one genre, 569 movies fall into two genres, 215 movies have three genres, and 65 belong in three or more genres. Using the Movies data, we also found that the oldest release date was January 1, 1922, corresponding to the movie *Nosferatu* and the newest release date was October 23, 1998 corresponding to the movie *Apt Pupil.*

In the ratings data, we found that most people rated movies with a three or four. The ratings distribution can be seen above in figure 6.

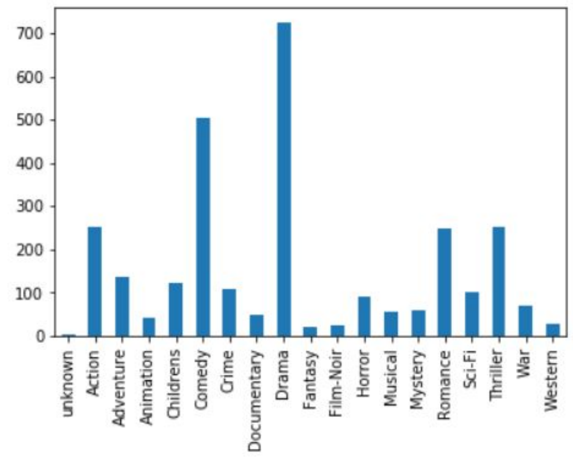


Figure 5: Genres Distribution

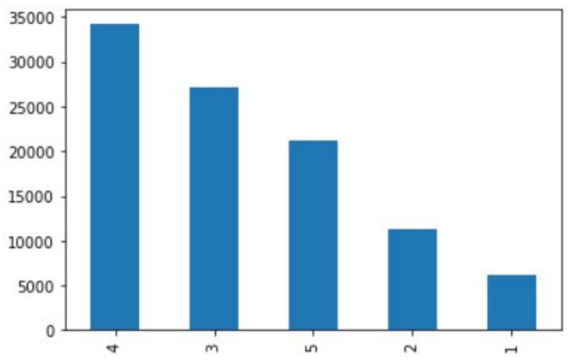
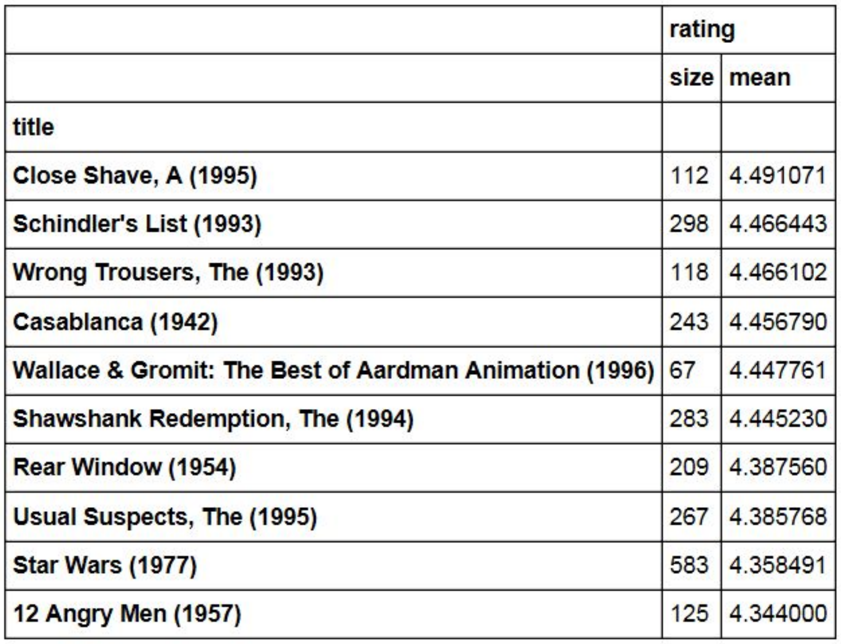


Figure 6: Ratings Distribution

Trends Found in Merged Data

After a preliminary analysis of each dataset, we merged the three files in python by the user id and movie id. We wanted to explore trends across the three files. One of our first questions was whether or not males and females tended to watch different genre movies. We created barcharts of genre counts by gender (seen in figures 7 and 8 below) , and found that while men have watched more from each genre over all, that is likely due to the unbalanced distribution. The shape of each bar plot for the genre counts is about the same. Females do tend to watch more romance movies, and males more thriller and sci-fi, though.

Next, we wanted to find what were the highest rated movies in the data set. Since some movies have only been rated by one user, we found that searching for the highest rated movies wasn't enough. We created a subset of the data that only includes movies that were rated 50 times or more. From that data set, we found that A Close Shave was the highest rated movie (mean rating of 4.49). See the top ten movies and their mean rating in figure 9 below. Note that the size column corresponds to the number of times the movie was rated.



	rating	
	size	mean
title		
Close Shave, A (1995)	112	4.491071
Schindler's List (1993)	298	4.466443
Wrong Trousers, The (1993)	118	4.466102
Casablanca (1942)	243	4.456790
Wallace & Gromit: The Best of Aardman Animation (1996)	67	4.447761
Shawshank Redemption, The (1994)	283	4.445230
Rear Window (1954)	209	4.387560
Usual Suspects, The (1995)	267	4.385768
Star Wars (1977)	583	4.358491
12 Angry Men (1957)	125	4.344000

Figure 9: Top Movies by Average Rating

We wanted to know which movies have the most ratings, ie, which movies were the most seen or most popular. We found that Star Wars, Contact, Fargo, and Return of the Jedi are some of the most seen movies. A table of the top watched movies can be seen in the appendix (item 1).

Next, we wanted to see if there were any initial trends in how different users rated movies based on their age, gender, or occupation. By created tables of the average ratings across these variables (seen in appendix items 2-4), we could form some conclusions. We found that there is a positive correlation between age and rating-- younger users have a lower average rating than the older users. We also see there isn't a significant difference between the average ratings for males (3.529) and females (3.531). Finally, we find that healthcare workers have the lowest average rating, at 2.89. Lawyers have the highest average rating, at 3.73.

Finally, we wanted to further explore how males and rate movies differently. Using the 50 most rates movies subset we created earlier, we made a pivot table of the mean rating of each of those movies by gender. We also created a column in the table that is the difference between the male”s mean rating and the females. This difference column is visualized in figure 10. From the visual above, we see that the movies that have the biggest rating differences among genders are "independence Day" (preferred by the women) and "Terminator" (preferred by the men). Males and females felt similarly about "Indiana Jones" and "Return of the Jedi".

Our initial analysis allows us to conclude the gender, age, and occupation all might be significant predictors of how an individual will rate a movie. From our last visualization, we see that males and females rate movies differently, so we expect genre to be a significant predictor as well.

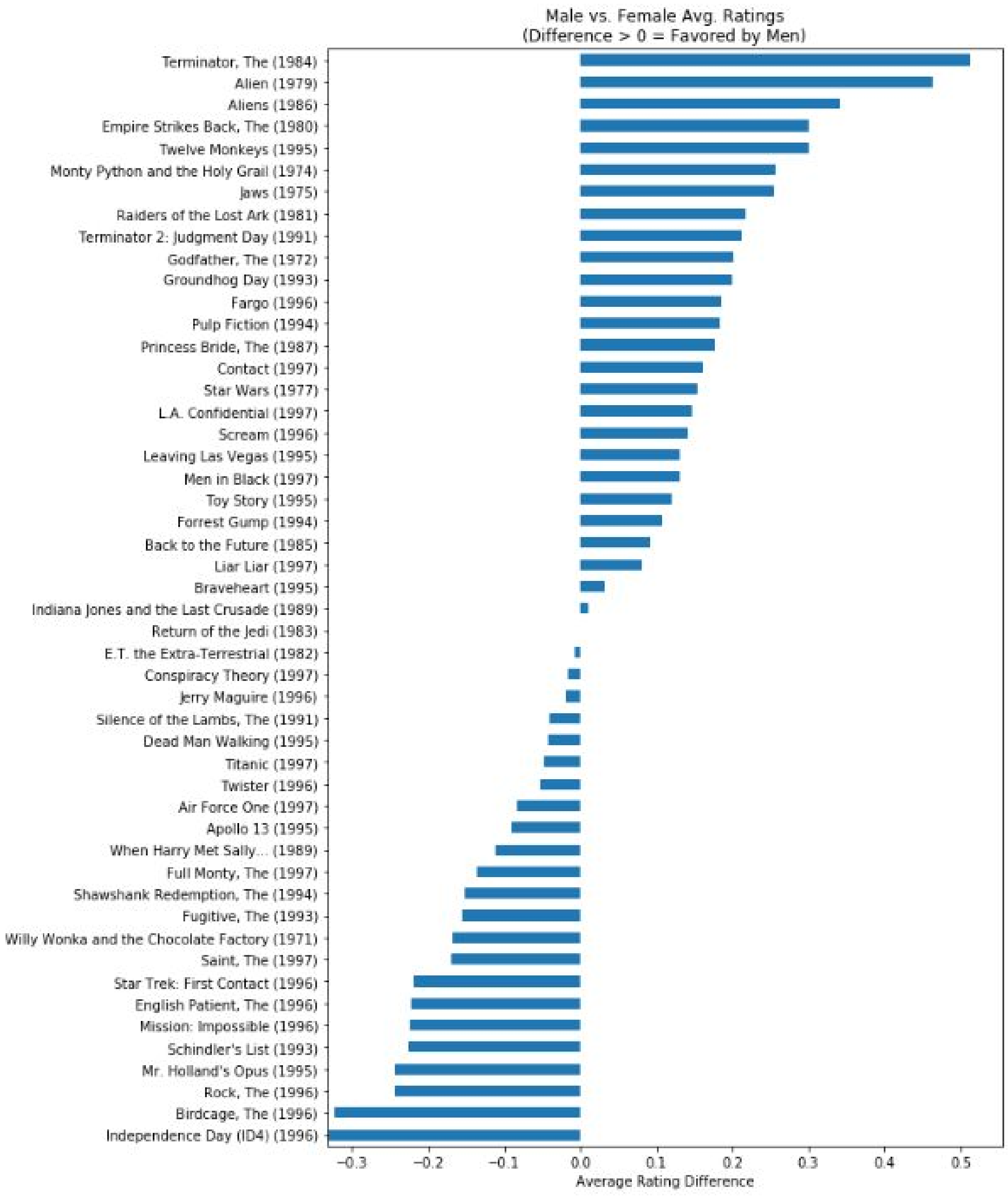
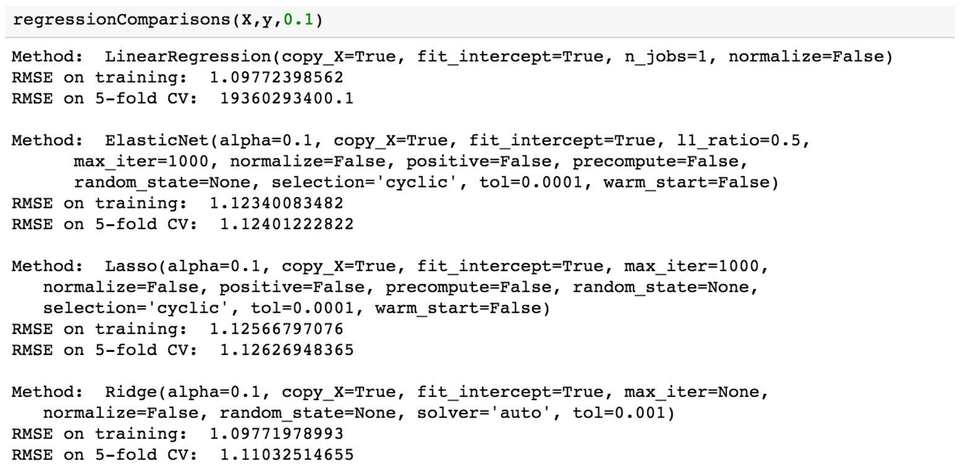


Figure 11: Difference in Ratings by Gender

Regression and Decision Trees

Multilinear Regression, Ridge Regression and Lasso Regression

After we completed the preliminary analysis, we decided a good first attempt would be linear regression. Perhaps a linear model of our other variables could be used to predict a move rating. We created a function that would generate different linear models for the data and print the results, seen in figure 12 below.



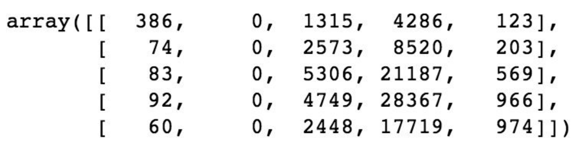
```
regressionComparisons(X,y,0.1)

Method:  LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
RMSE on training:  1.09772398562
RMSE on 5-fold CV:  19360293400.1

Method:  ElasticNet(alpha=0.1, copy_X=True, fit_intercept=True, l1_ratio=0.5,
      max_iter=1000, normalize=False, positive=False, precompute=False,
      random_state=None, selection='cyclic', tol=0.0001, warm_start=False)
RMSE on training:  1.12340083482
RMSE on 5-fold CV:  1.12401222822

Method:  Lasso(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=1000,
   normalize=False, positive=False, precompute=False, random_state=None,
   selection='cyclic', tol=0.0001, warm_start=False)
RMSE on training:  1.12566797076
RMSE on 5-fold CV:  1.12626948365

Method:  Ridge(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001)
RMSE on training:  1.09771978993
RMSE on 5-fold CV:  1.11032514655
```

Figure 12

As you can see, Ridge regression was the most successful with a RMSE of 1.09. However, those results are still not very accurate. Since our data is ratings, an ordinal variable from 1 to 5, it might be better to consider the y variable a category.

Logistic Regression

Next, we attempted a logistic regression. We trained the model such that it would consider the 1-5 ratings as five independent binary outcomes. Unfortunately, this model was also unsuccessful. The confusion matrix is as follows:



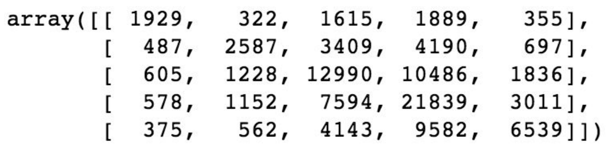
```
array([[  386,      0,  1315,  4286,    123],
       [   74,      0,  2573,  8520,    203],
       [   83,      0,  5306, 21187,    569],
       [   92,      0,  4749, 28367,    966],
       [   60,      0,  2448, 17719,    974]])
```

Figure 13

The model had an accuracy of 35%.

Decision Tree

Finally, we attempted a decision tree. It was more successful than our linear our regressions. Its accuracy was 46% and the confusion matrix was as follows:



```
array([[ 1929,   322,  1615,  1889,   355],
       [  487,  2587,  3409,  4190,   697],
       [  605,  1228, 12990, 10486,  1836],
       [  578,  1152,  7594, 21839,  3011],
       [  375,   562,  4143,  9582,  6539]])
```

Figure 14:

As you can see, this model has much more success, particularly for movies with a 1 rating or a 5 rating. However, these models did not give us much insight on what determines a movie rating. Based on our results thus far, we decided to move on to clustering to gain a better understanding of the movies and users.

Clustering

kmeans

Clustering was done on the entire dataset to determine which movies and genres are distinct from each other. Based on the results, we could also determine which users were representative in each cluster.

For kmeans, distance metrics used are cosine, eucidean and cityblock. Elbow method was used to determine the optimal value of k. Plots are shown below:

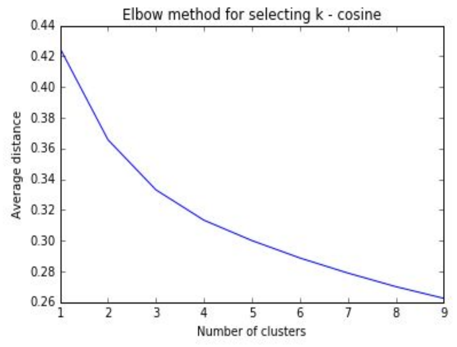


Figure 15 cosine – Elbow plot

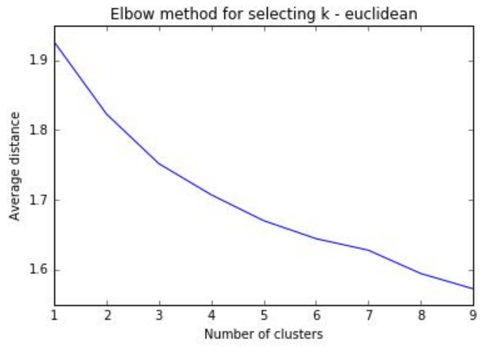


Figure 16 Euclidean – Elbow plot

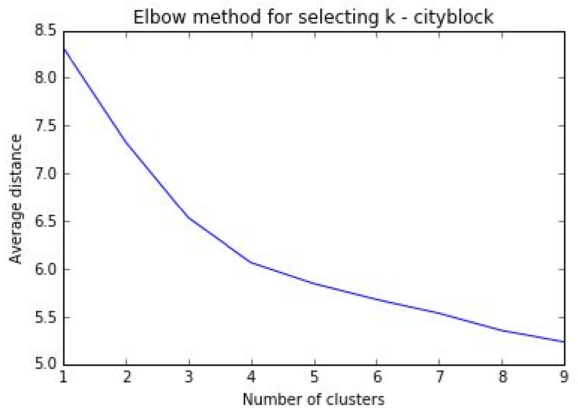


Figure 17 Cityblock - Elbow plot

From the plots in figure 17, we cannot see an apparent knee point to determine the value of k. For this reason, we compared the results of different distance metrics with value of k = 3 and 5 and finally used Euclidean distance as the distance metric for kmeans algorithm with k =5. Results are shown below:

TOP 5 GENRES:

Drama, Comedy and Action are represented most in all clusters. The dataset contains movies from these 3 genres the most as seen from the preliminary analysis. Bar plots of top 5 genres in each cluster are shown in figures 18, 19, 20, 21, 22

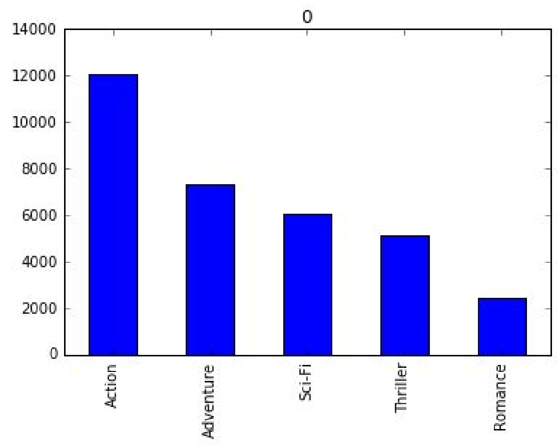


Figure 18 Top 5 genre Cluster 0

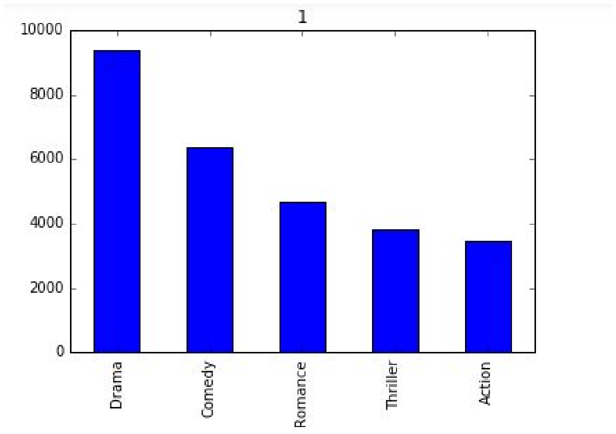


Figure 19 Top 5 genre Cluster 1

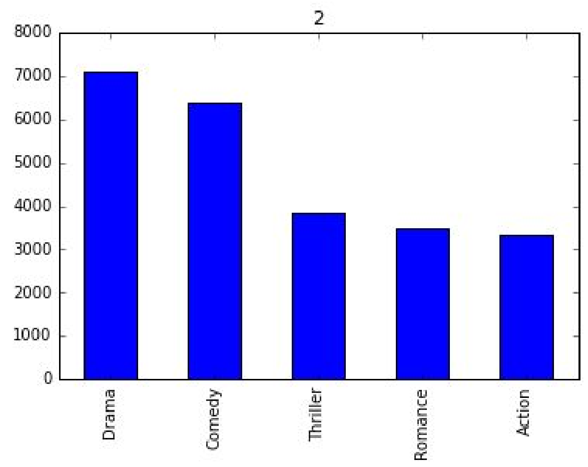


Figure 20 Top 5 genres - Cluster 2

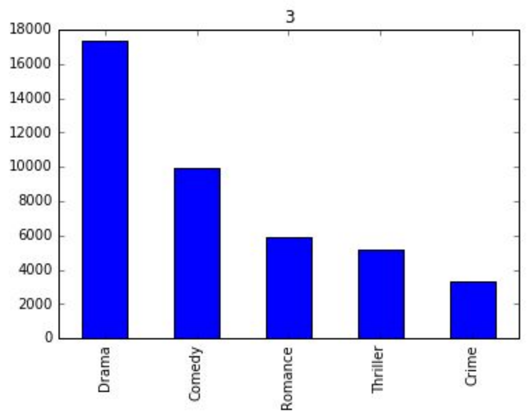


Figure 21 Top 5 genres - Cluster 3

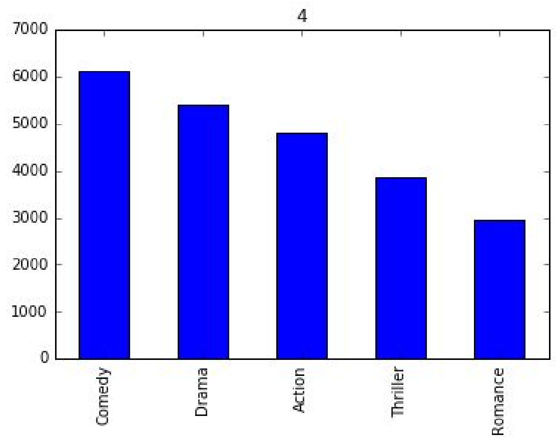
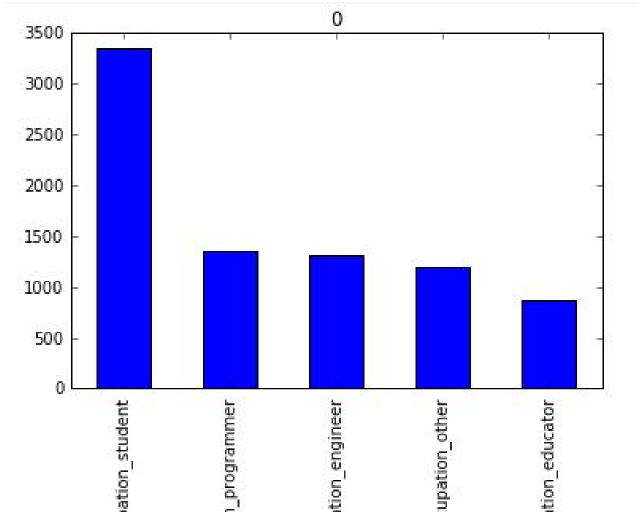


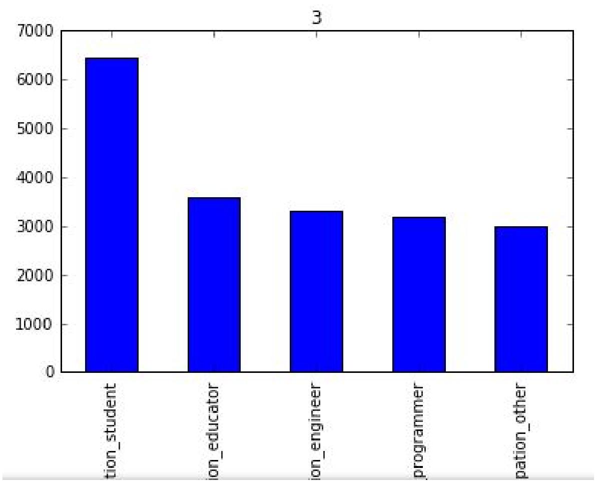
Figure 22 Top 5 genres Cluster 4

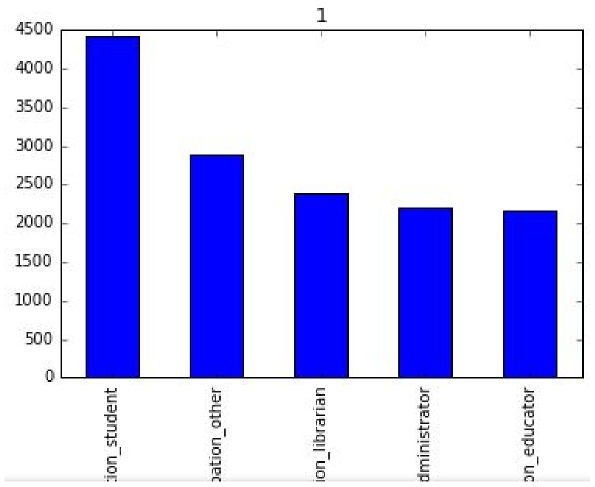
TOP 5 OCCUPATIONS:

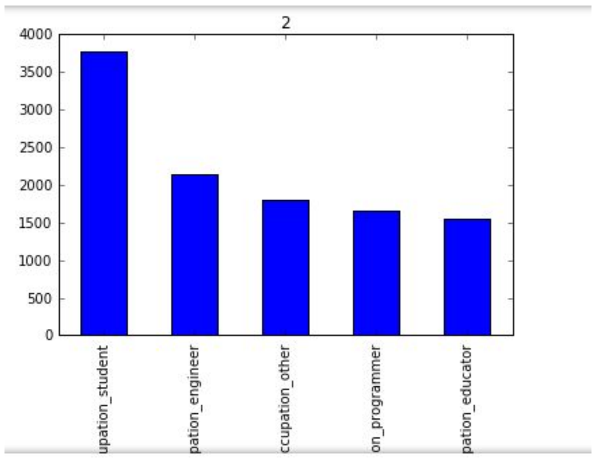
Students form core of the user subset in each cluster. Occupation represented in each cluster is summarized below:

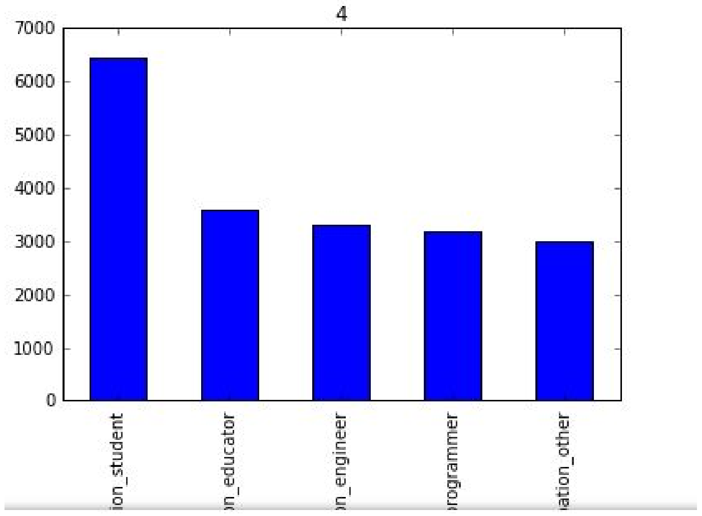
Cluster 0 – programmers and engineers
Cluster 1 – others and librarians
Cluster 2 – engineers and others
Cluster 3 – educators and engineers
Cluster 4 – educators and engineers



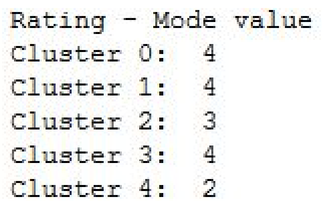








Rating in each cluster is shown in figure 23. Clusters 0, 1, and 3 are represented most with rating 4 whereas clusters 2 and 4 are represented by rating 3 and 2 respectively.



```
Rating - Mode value
Cluster 0:  4
Cluster 1:  4
Cluster 2:  3
Cluster 3:  4
Cluster 4:  2
```

Figure 23 Rating - Mode value

Results:

Cluster 0 – Students, programmers and engineers that like movies from the genres action, adventure and scifi with average rating of 4.

Cluster 1 – Students and librarians that like movies from the genres drama, comedy and romance with average rating of 4.

Cluster 2 – Students and engineers that like movies from the genres drama, comedy and thriller with average rating of 3.

Cluster 3 – Students, educators and engineers that like movies from the genres drama, comedy and romance with average rating of 4.

Cluster 4 – Students, educators and engineers that like movies from the genres drama, comedy and action with average rating of 2.

Hierarchical Agglomerative Clustering

This method was time consuming and took more than an hour to run on the training dataset (owing to the size of the dataset 18MB). I am attaching code for this method using test dataset. It was used only to understand how Hierarchical Clustering is done and, how to plot dendrograms in python.

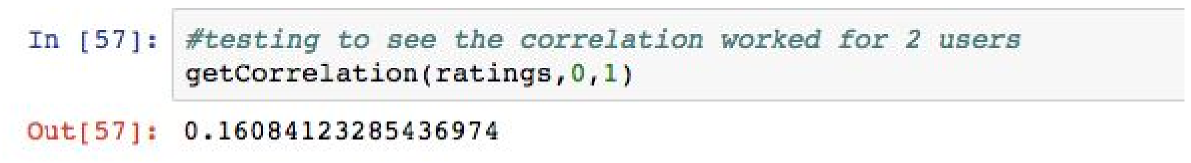
Recommender System

Method

We wanted to make a recommender system that takes in a user_id and gives them 3 movie_id's as recommendations to watch. We loaded in our data that had the users and all the movies they had rated. From this dataset we were able to get all the unique users and all the unique movies. We then made an array that included all the users and the movie and the ratings for each user to the movie that they have watched (1-5). If they hadn't rated a movie there was a 0 there.

We were then able to get the correlations with a getCorrelation method for any 2 different users. This method takes our data set and 2 user ids whom we want to see the correlation with. It then calculates the correlation between a movie that they have both rated. If there is no movie they have both rated it will return 0. If the correlation is negative it will also return 0. Otherwise it will return the correct correlation between the 2 users.

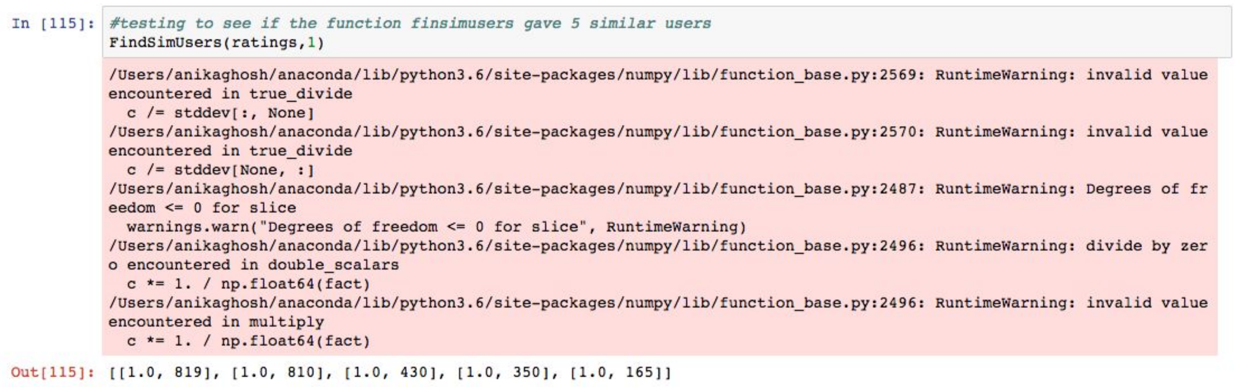
We tested this on user_id 0 and 1 to get the correlation



```
In [57]: #testing to see the correlation worked for 2 users
         getCorrelation(ratings,0,1)

Out[57]: 0.16084123285436974
```

After we made function called FindSimUsers that allows you to get 5 similar users when putting in one user_id called findsimusers. This uses the above method getCorrelation to spit out the 5 users. It finds the correlation of that user with everyone, and then gives you the top 5 with the highest correlations.

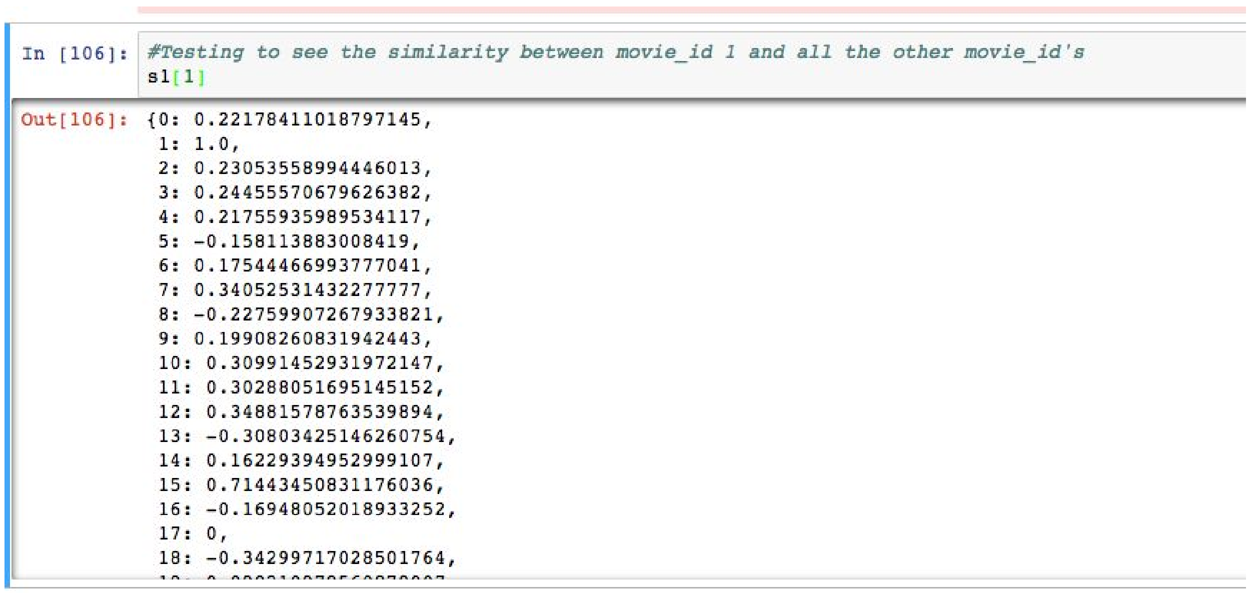
We tested this on user 1 and saw the top 5 users it is most similar with.



```
In [115]: #testing to see if the function finsimusers gave 5 similar users
          FindSimUsers(ratings,1)

          /Users/anikaghosh/anaconda/lib/python3.6/site-packages/numpy/lib/function_base.py:2569: RuntimeWarning: invalid value
          encountered in true_divide
            c /= stddev[:, None]
          /Users/anikaghosh/anaconda/lib/python3.6/site-packages/numpy/lib/function_base.py:2570: RuntimeWarning: invalid value
          encountered in true_divide
            c /= stddev[None, :]
          /Users/anikaghosh/anaconda/lib/python3.6/site-packages/numpy/lib/function_base.py:2487: RuntimeWarning: Degrees of fr
          eedom <= 0 for slice
            warnings.warn("Degrees of freedom <= 0 for slice", RuntimeWarning)
          /Users/anikaghosh/anaconda/lib/python3.6/site-packages/numpy/lib/function_base.py:2496: RuntimeWarning: divide by zer
          o encountered in double_scalars
            c *= 1. / np.float64(fact)
          /Users/anikaghosh/anaconda/lib/python3.6/site-packages/numpy/lib/function_base.py:2496: RuntimeWarning: invalid value
          encountered in multiply
            c *= 1. / np.float64(fact)

Out[115]: [[1.0, 819], [1.0, 810], [1.0, 430], [1.0, 350], [1.0, 165]]
```

We then made a variable called GetSimUsers called that gets the correlations between each movie so you could put down one move and see the correlation between that and all of the other movies based on a similarity measure. The similarity measure we used was PearsSimilarity. It finds the correlation by calculating the correlations of the ratings of each movie. We named a variable called s1 that holds all the correlations. We can can that use that variable to find the correlations with every movie.

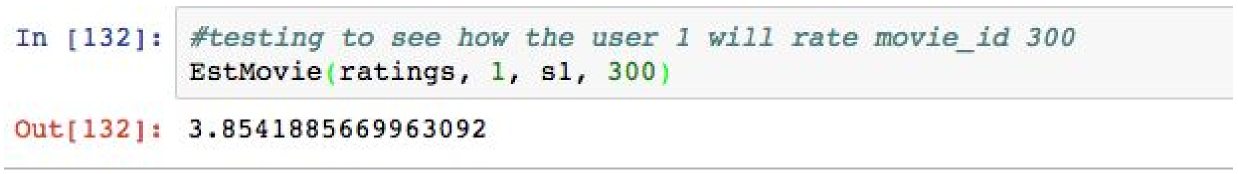
We tested it below with movie 1:



```
In [106]: #Testing to see the similarity between movie_id 1 and all the other movie_id's
          s1[1]

Out[106]: {0: 0.22178411018797145,
           1: 1.0,
           2: 0.23053558994446013,
           3: 0.24455570679626382,
           4: 0.21755935989534117,
           5: -0.158113883008419,
           6: 0.17544466993777041,
           7: 0.34052531432277777,
           8: -0.22759907267933821,
           9: 0.19908260831942443,
           10: 0.30991452931972147,
           11: 0.30288051695145152,
           12: 0.34881578763539894,
           13: -0.30803425146260754,
           14: 0.16229394952999107,
           15: 0.71443450831176036,
           16: -0.16948052018933252,
           17: 0,
           18: -0.34299717028501764,
```

We then made a function EstMovie and within this function it finds the movies the user hasn't rated yet and then tests those similarities with the movies the user has rated and gives the ratings the user has given to those movies.   returns the predicted rating of the movie the users hasn't rated is similar to the movies she has rated high. It does this by using the above function to get the correlations between each movie and then based on those correlations with the movies the user has seen, it can predict the ratings of every movie he/she hasn't seen.

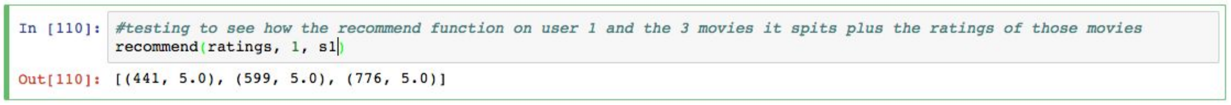
We tested this with user 1 and movie id 300. We wanted to see how user 1 will rate movie id 300.



```
In [132]: #testing to see how the user 1 will rate movie_id 300
          EstMovie(ratings, 1, s1, 300)

Out[132]: 3.8541885669963092
```

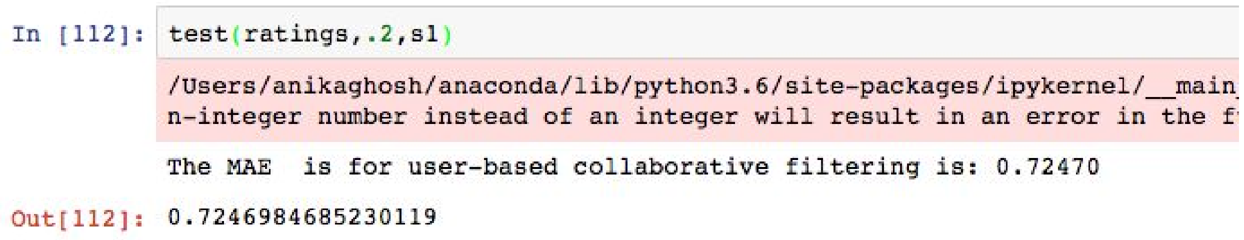
We then made a recommend function that uses the function above that rates each movie for the user. It finds the list of unrated items for the given user and estimates its score. It then spits out the number that given amount of movies to watch as a recommendation. The function sorts it from highest to lowest so it only gives the movies with the highest scores.

We tested this on user 10 and it gave us back 3 movies with how much user 10 would rate them



```
In [110]: #testing to see how the recommend function on user 1 and the 3 movies it spits plus the ratings of those movies
          recommend(ratings, 1, s1)

Out[110]: [(441, 5.0), (599, 5.0), (776, 5.0)]
```

We wanted to test how accurate our recommendation is so we made a function called cross_validate_user that calculated all the error from the items that we predicted and then used this in another function called test to calculate the Mean Average Error for all of our predicted items. The cross_validate_user does this by splitting the data and using part of the data for testing. It then makes a new data frame of the users and rated items in the test data set. It then rates the movies based on the EstMovie method that we had created and then compares these ratings to the given ratings we already had. It calculates the error by doing the abs value (estimated score - actual score). The test function then uses the above calculations and error and count rate for every user. It calculates the MAE by doing the total error/ total count.



```
In [112]: test(ratings,.2,s1)
          /Users/anikaghosh/anaconda/lib/python3.6/site-packages/ipykernel/__main
          n-integer number instead of an integer will result in an error in the f

          The MAE  is for user-based collaborative filtering is: 0.72470
Out[112]: 0.7246984685230119
```

We got a MAE score of of around .724. This isn't that great of a score, but a reason it could be so high is that there were only 5 ratings a person could do. This number could be lower using the same functions had the rating scale been 1-10 or even better had it been 1-100.

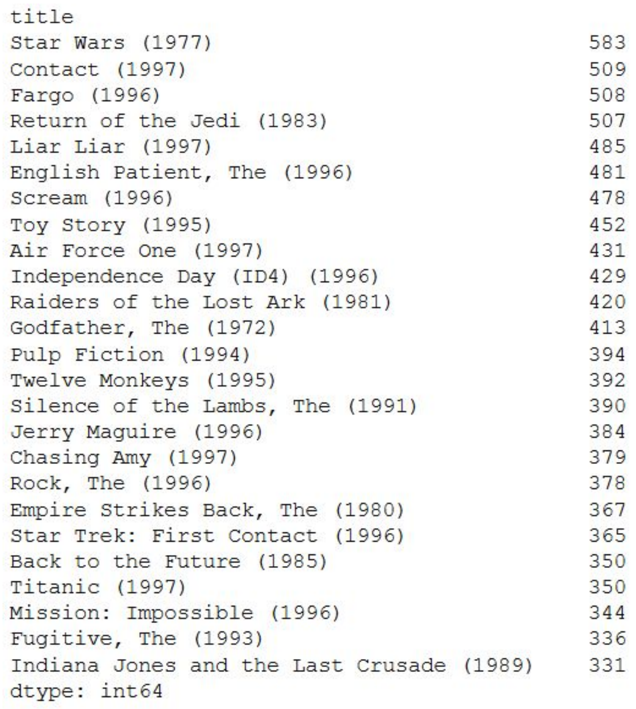
Conclusion

With this project, we found that many factors influence a movie rating. Though linear regressions were not successful in modeling our data, decisions trees proved to be better. In the future, it might be worthwhile to expand our initial model into a random forest.

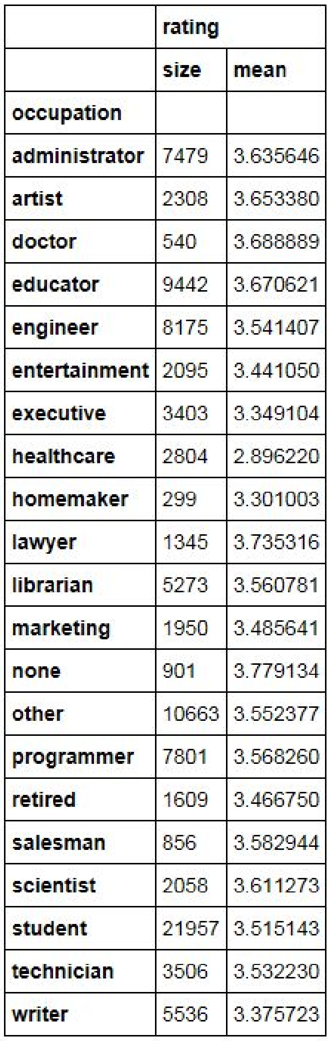
We also found that clustering was appropriate for this data set. We used kmeans clustering to determine inherent grouping of movies and users. We found five distinct clusters within the data, each which had different trends in movie genre, and user occupation in particular.

Finally, we created a recommender system that given a user ID, can recommend movies the user hasn't seen but is expected to score highly. We used user similarity and movie similarity for this process. Though we had significant error in our final model, this was a great first attempt for a recommender system and we learned a lot in the process.

Appendix



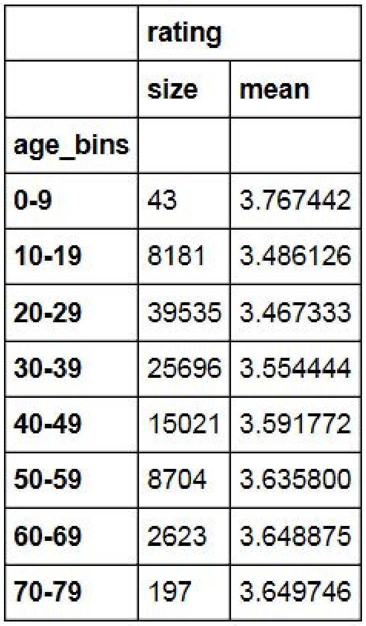
```
title
Star Wars (1977)                                 583
Contact (1997)                                   509
Fargo (1996)                                     508
Return of the Jedi (1983)                        507
Liar Liar (1997)                                 485
English Patient, The (1996)                      481
Scream (1996)                                    478
Toy Story (1995)                                 452
Air Force One (1997)                             431
Independence Day (ID4) (1996)                    429
Raiders of the Lost Ark (1981)                   420
Godfather, The (1972)                            413
Pulp Fiction (1994)                              394
Twelve Monkeys (1995)                            392
Silence of the Lambs, The (1991)                 390
Jerry Maguire (1996)                             384
Chasing Amy (1997)                               379
Rock, The (1996)                                 378
Empire Strikes Back, The (1980)                  367
Star Trek: First Contact (1996)                  365
Back to the Future (1985)                        350
Titanic (1997)                                   350
Mission: Impossible (1996)                       344
Fugitive, The (1993)                             336
Indiana Jones and the Last Crusade (1989)        331
dtype: int64
```



	rating	
	size	mean
occupation		
administrator	7479	3.635646
artist	2308	3.653380
doctor	540	3.688889
educator	9442	3.670621
engineer	8175	3.541407
entertainment	2095	3.441050
executive	3403	3.349104
healthcare	2804	2.896220
homemaker	299	3.301003
lawyer	1345	3.735316
librarian	5273	3.560781
marketing	1950	3.485641
none	901	3.779134
other	10663	3.552377
programmer	7801	3.568260
retired	1609	3.466750
salesman	856	3.582944
scientist	2058	3.611273
student	21957	3.515143
technician	3506	3.532230
writer	5536	3.375723

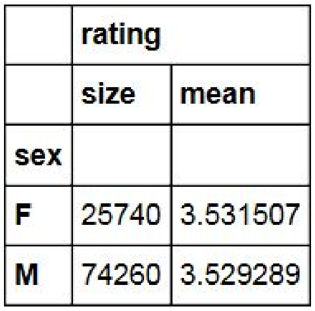
appendix item 1: movies with most ratings

appendix item 2: rating by occupation



	rating	
	size	mean
age_bins		
0-9	43	3.767442
10-19	8181	3.486126
20-29	39535	3.467333
30-39	25696	3.554444
40-49	15021	3.591772
50-59	8704	3.635800
60-69	2623	3.648875
70-79	197	3.649746

appendix item 3



	rating	
	size	mean
sex		
F	25740	3.531507
M	74260	3.529289

appendix item 4