CIS 6930-Introduction to Data Mining
Group Project Report

Analysis on IMDB Dataset

Group Members

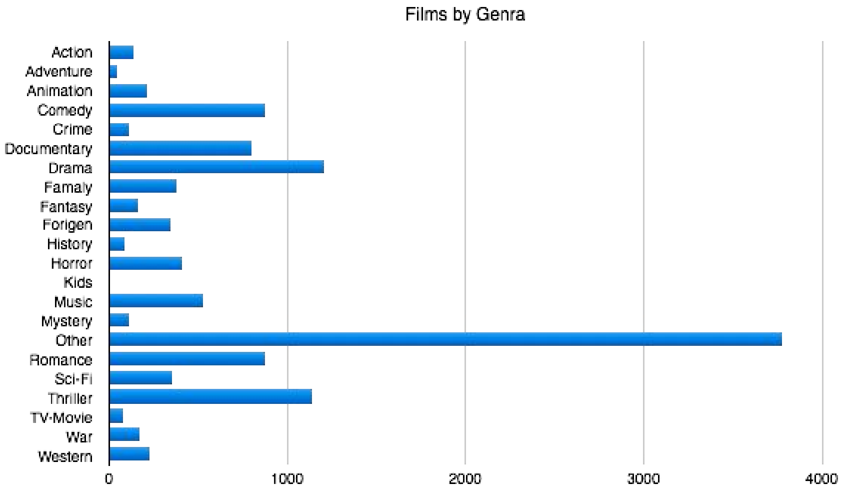
Nick Topper
Josh Campbell
Karthik Maharajan
Ankit Kulkarni
Arvindh Seshadhri

Literature Review

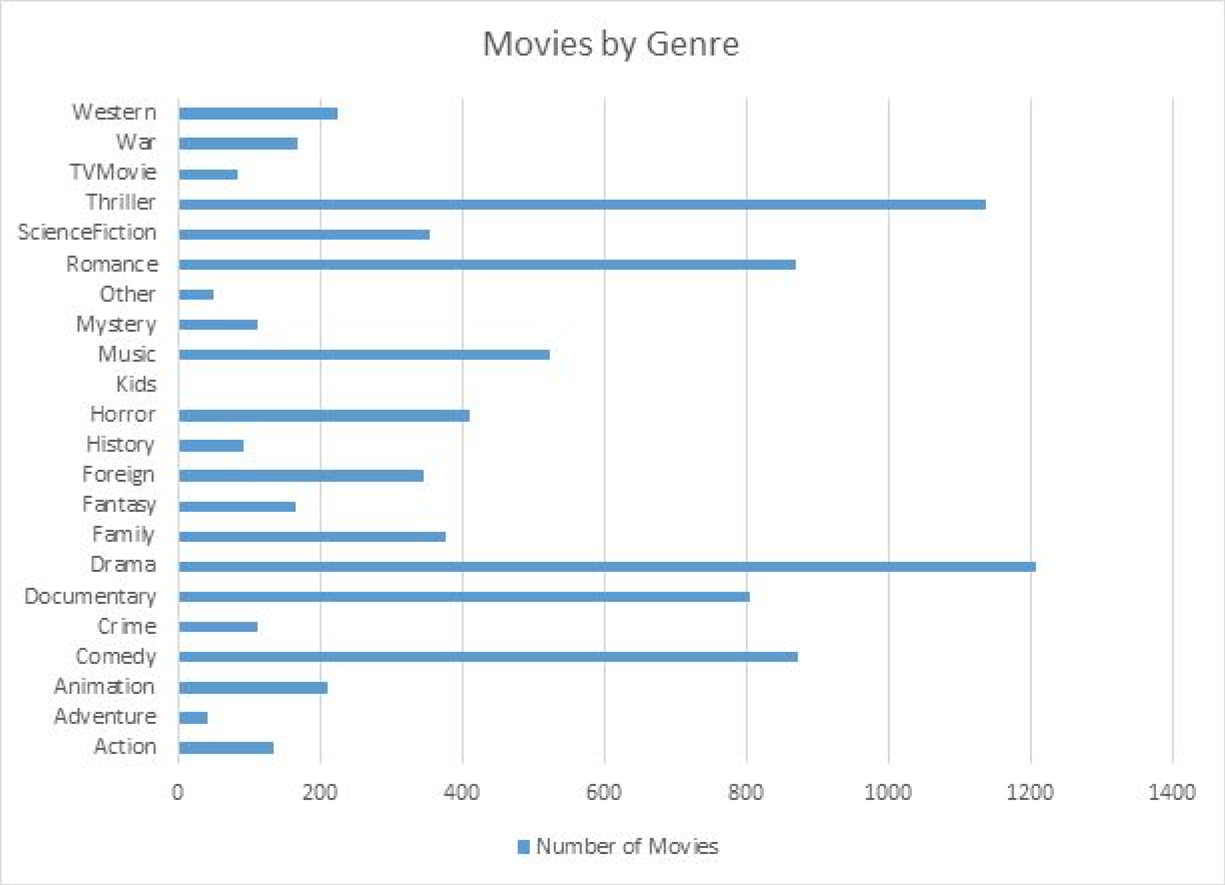
Most of the other past data mining work on the IMDB data were in connection with the netflix challenge[11]. The netflix challenge was concerned with predicting user ratings for movies based on previous ratings. Though these works were not closely related to the current project we leveraged few ideas from [1], [12] and [13]. [12] gave insights into the use of neighborhood models in the mining algorithms. [1] let us explore the application of naive bayes and K-means on the IMDB features. Based on our results naive bayes proved to be the best classification method for the IMDB dataset while K-means did not provide meaningful clusters. [13] provided some ideas for feature extraction. Since the results of these works mainly focussed on user rating prediction comparing their results with the current project is not feasible.

Statistics about the dataset

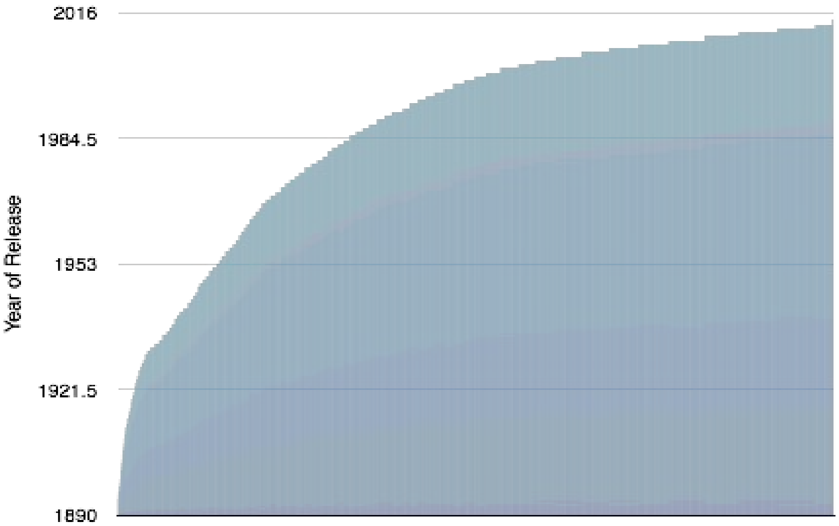
The final dataset contains 12035 films collected from TheMoviedb.org. The dataset includes movies belonging to more than 600 different genres (with movies having more than one sub genres). After processing the dataset to include only the main genre (in order to get better results), we have 23 different genres.



As suggested during the review, we created an unbiased sampling of the dataset which looks as follows, hoping to get better results.



Movies by Genre
Western
War
TVMovie
Thriller
ScienceFiction
Romance
Other
Mystery
Music
Kids
Horror
History
Foreign
Fantasy
Family
Drama
Documentary
Crime
Comedy
Animation
Adventure
Action
0
200
400
600
800
1000
1200
1400
■ Number of Movies



2016
1984.5
1953
1921.5
1890
Year of Release

Methods and Materials

Data Set Generation

Because of limited options for third-party data collection tools on the IMDB data set, data was collected from *themoviedb.org.* An API key was obtained, and a python script was written to obtain data from the *"themoviedb.org"* API. The python script selected 15000 movie Id numbers at random. Of these, roughly 13000 were used, the others discarded due to various HTTP and parsing errors. Of the films examined, the title, date of release, top 3 cast members (in order) and top 3 crew members (in order), as well as the list of genres were recorded. Due to inconsistencies in job titles given in the film industry, we chose to classify cast and crew members by the order that they appear in the credits, rather than the title. We chose to rely on the convention that cast and crew members with greater creative authority appear first on the credits. Choosing these features allows us to inspect the most important creative authorities, in order, on each side of the camera. We also processed the date of release feature to show only the year movie was released in.

Classification

In this project, we try to predict the genre of movies based on the other features in the dataset. We use the following classification algorithms to predict the genre,

Naive Bayes
C4.5 (Decision Tree)
RIPPER (Decision Tree)
Oblique tree (Decision Tree)
K-nearest neighbor

These are being run using CRANs implementations in the RWeka, Caret, and e1071 packages.

We will run each classification algorithm on the data set targeting Genre as the classification variable. We will then access the Confusion Matrix for each rule to get values indicating the accuracy of each prediction algorithm. Comparing these values we can determine which classification algorithm produces the best results for this dataset.

After our first pass, we found that we had far too specific genres to produce meaningful results, so we spent some time altering the data set to have less specific genres to allow for more meaningful and accurate classification analysis. This will be explained further in the results section.

Association Mining

The implementation of the Apriori algorithm found in CRAN's 'arules' package was used to generate association rules. Two separate R-scripts were created, one to generate rules associating actors to co-stars and high-ranking crew members, and another to generate rules associating actors to genres.

In order to discover rules associating actors to co-stars and crew members, a new table was created from the existing dataset. For each actor in each film in the original dataset, a new row was created. Each row contains an actor, their two most significant co-stars, and their three most significant crew members for each film that they participated in. Apriori was used with a minimum support of 0.0002 and no minimum confidence. Of the top 50 rules whose right-hand-side contained "actor", sorted by lift, the top 20 non-redundant rules were reported.

Clustering

Here we use the ROCK (RObust Clustering using linKs) Clustering algorithm to cluster the values and attributes in the dataset based on the number of genres. The traditional methods such as K-Means and DBSCAN could not be used here because the dataset contains both nominal and numeric values. The use of the K-Modes clustering was tried but it was dropped because, even though it could be used for this dataset, the running time was too high in order to get a successful implementation.

The clustering was done based on the number of genres present in the dataset. ROCK Clustering redefines the distances between points to be the number of shared neighbors whose strength is greater than a given threshold. It then uses a hierarchical clustering scheme to cluster the data. ROCK clustering gives better quality of clustering compared to other traditional algorithms due to the usage of links, and so this was one of the best algorithms which could be used in this dataset.

Results and Discussion

Classification

Our classification results took two iterations. At first we were classifying using very specific genres that took into account both the main genre of the film and every single sub-genre listed for it as one classifier. We found that even with 13000 entries, with genres this specific there might still only be two to three films with that genre, leading to our poor results. As one might expect this led to our methods producing results that were not very meaningful, with very low accuracy, precision, recall, and f-measure for every algorithm we attempted to classify the data set with.

After facing these poor results we decided to alter the data set so that we could get more meaningful results. As a result we decided to prune the dataset and only take into account the main genre assigned to each film. For example, we would take a film whose genre had previously been “Action-Adventure-Drama-ScienceFiction-War-Thriller” and assign it “Thriller”. This cut our classification options from well over 100 incredibly specific genres to 23 more broad, but still descriptive genres.

Naive Bayes

We found that Naive Bayes gave us the best results of the algorithms that we ran. Looking at the confusion matrix, we get a print out of the following results:

```
Accuracy : 0.3477
95% CI : (0.3287, 0.3671)
No Information Rate : 0.3265
P-Value [Acc > NIR] : 0.01442
Kappa : 0.1222
Mcnemar's Test P-Value : NA
```

Most importantly this tells us that Naive Bayes predicts with an accuracy of 34.77% for this data set.

The fact that Naive Bayes is the best performing algorithm is likely due to the nature of the data. Naive Bayes treats each attribute as independent, instead of as parts of a whole, and each attribute contributes to the probability of a classification instead of guarantees it. Similarly, with our data, a movie starring say Samuel L Jackson and Nicolas Cage does not make it an action film, but might make it have a higher probability of being an action film.

The genre ‘Other’ played a huge role in accuracy of Classification in our dataset. We created a sub-sample of the original dataset, reducing the number of movies classified as ‘Other’ to tackle this issue. The results achieved were as follows:

```
Accuracy : 0.1805
95% CI : (0.1623, 0.1999)
No Information Rate : 0.1336
P-Value [Acc > NIR] : 4.229e-08
Kappa : 0.095
Mcnemar's Test P-Value : NA
```

The accuracy achieved for this dataset is 18.05% which indicates that sub-sampling did not improve the results of Naive Bayes.

C4.5

Though C4.5 produces very weak results it is still the second most feasible, as it completed the classification in a reasonable amount of time. The results it produced are as follows:

```
Accuracy : 0.0798
95% CI : (0.0693, 0.0913)
No Information Rate : 0.3265
P-Value [Acc > NIR] : 1
Kappa : 1e-04
Mcnemar's Test P-Value : NA
```

The genre ‘Other’ played a huge role in accuracy of Classification in our dataset. We created a sub-sample of the original dataset, reducing the number of movies classified as ‘Other’ to tackle this issue. The results achieved were as follows:

```
Accuracy : 0.133
95% CI : (0.117, 0.1502)
No Information Rate : 0.1336
P-Value [Acc > NIR] : 0.5396
Kappa : 3e-04
Mcnemar's Test P-Value : NA
```

The accuracy achieved for this dataset is 13.3% which indicates that sub-sampling improved the results of C4.5.

RIPPER

RIPPER proved to be the third most feasible. Unlike Oblique it was indeed working, indicating that it could be successfully used to classify a dataset of this size. However after letting it run for roughly 60 minutes, just in an attempt to build a training set, we realized that it was not a truly feasible option for this data, primarily due to the sheer size of the set. This is to be expected as the time RIPPER takes grows linearly with set size. Since we have a huge set, RIPPER takes a huge amount of time.

Oblique Tree

After attempting to run the oblique.tree algorithm we found that it was not a good fit for this data set. In fact it proved to be impossible to use for this set. The function used to implement this algorithm always returned an error as follows:

```
Error in 1:(2^(number.of.residual.factors - 1) - 1) :
                    result would be too long a vector
```

This indicates that the data set is far too large for the oblique tree algorithm to be feasible as a classification method, and we should not use it for this data set.

K-Nearest Neighbor

This was another algorithm that did not work for our dataset. Due to the size of the data set it is expected that there may be some entry that is missing a value. This turned out to be true, thus making the KNN classification algorithm unusable.

General Discussion for Classification

Though some of the algorithms worked better than others, we can see that in general the algorithms had difficulty correctly classifying the movies in our data set based on the attributes we presented. This is likely due to the attributes we chose in conjunction with the size of the set, and the size of the overall set that we drew from. Our attributes consist of information such as title, year, and several of the most important cast and crew members. Obviously most films will have different titles, and with the number of films released each year, it is easy to see that there is not much, if any, effect that year would have on the genre of a film. This leaves the cast and crew, which are theoretically the most meaningful attributes, as actors and directors tend to take part in similar films. However, with the size of the data set we are drawing from, having millions of movies from all over the world, even with a data set of size 13000, there is no guarantee that we will have many movies with the same actors or directors or such, let alone enough of the same genre to allow for successful classification. This is likely to blame for the low success rate of even our most accurate classification algorithm.

Association Mining

The Usual Casts

Rule	Support	Conf	Lift
{costar1= David Lochary} => {Actor= Divine}	0.0002530364	1	3952
{costar2= David Lochary} => {Actor= Mary Vivian Pearce}	0.0002530364	1	3952
{costar1= David Lochary,costar2= Mary Vivian Pearce} => {Actor= Divine}	0.0002530364	1	3952
{costar1= David Lochary,crew2= John Waters} => {Actor= Divine}	0.0002530364	1	3952
{costar1= Divine,costar2= David Lochary} => {Actor= Mary Vivian Pearce}	0.0002530364	1	3952
{costar2= David Lochary,crew1= John Waters} => {Actor= Mary Vivian Pearce}	0.0002530364	1	3952
{costar1= Divine,costar2= Mary Vivian Pearce} => {Actor= David Lochary}	0.0002530364	1	3952
{costar1= Joe Penny,costar2= William R. Moses} => {Actor= Lea Thompson}	0.0002530364	1	3952
{costar1= David Lochary,costar2= Mary Vivian Pearce,crew1= John Waters} => {Actor= Divine}	0.0002530364	1	3952
{costar1= David Lochary,crew1= John Waters,crew3= John Waters} => {Actor= Divine}	0.0002530364	1	3952
{costar1= Divine,costar2= David Lochary,crew1= John Waters} => {Actor= Mary Vivian Pearce}	0.0002530364	1	3952
{costar1= Divine,costar2= Mary Vivian Pearce,crew1= John Waters} => {Actor= David Lochary}	0.0002530364	1	3952

{costar1= David Lochary,costar2= Mary Vivian Pearce,crew1= John Waters,crew3= John Waters} => {Actor= Divine}	0.0002530364	1	3952
{costar1= David Lochary,crew1= John Waters,crew2= John Waters,crew3= John Waters} => {Actor= Divine}	0.0002530364	1	3952
{costar1= Divine,costar2= David Lochary,crew2= John Waters,crew3= John Waters} => {Actor= Mary Vivian Pearce}	0.0002530364	1	3952
{costar2= David Lochary,crew1= John Waters,crew2= John Waters,crew3= John Waters} => {Actor= Mary Vivian Pearce}	0.0002530364	1	3952
{costar1= Divine,costar2= Mary Vivian Pearce,crew1= John Waters,crew2= John Waters} => {Actor= David Lochary}	0.0002530364	1	3952
{costar1= Divine,costar2= Mary Vivian Pearce,crew1= John Waters,crew2= John Waters,crew3= John Waters} => {Actor= David Lochary}	0.0002530364	1	3952
{costar1= David Lochary,costar2= Mary Vivian Pearce,crew1= John Waters,crew2= John Waters,crew3= John Waters} => {Actor= Divine}	0.0002530364	1	3952
{costar1= Divine,costar2= David Lochary,crew1= John Waters,crew2= John Waters,crew3= John Waters} => {Actor= Mary Vivian Pearce}	0.0002530364	1	3952

Relation between Genre and Cast

The apriori algorithm produced the following rules indicating relationships between cast members and the movie genre. The rules are sorted based on their lift values.

Rule	Support	Conf	Lift
{crew1= Walt Disney} => {genre= Animation}	0.001080273	0.8666667	53.211565
{genre= Animation} => {cast1= Clarence Nash}	0.001911251	0.1173469	50.434038
{crew2= Walt Disney} => {genre= Animation}	0.001661958	0.8000000	49.118367
{crew1= Jules White} => {genre= Comedy}	0.001246468	0.9375000	15.163810
{cast1= Curly Howard} => {genre= Comedy}	0.001163370	0.9333333	15.096416
{cast1= Mel Blanc} => {genre=Family}	0.001163370	0.2916667	13.818570

{cast1= Vijayakanth} => {genre= Drama}	0.001080273	0.7647059	12.237328
{crew1= Ilayaraja} => {genre= Drama}	0.001661958	0.6451613	10.324296
{crew1= Robby D.} => {genre=Other}	0.001163370	0.8235294	2.625259
{crew1= Dave Fleischer} => {genre=Other}	0.001495762	0.6206897	1.978644
{cast1= Mel Blanc} => {genre=Other}	0.001329566	0.3333333	1.062605

The Apriori algorithm produced the following rules indicating relationships between cast members and the movie genre after subsampling the original dataset to remove the bias towards movies with genre - “Other”. The subsampling improved the association mining results since the number of rules increased to 17 as shown below. The rules are sorted based on their lift values.

Rule	**Support**	**Confidence**	**Lift**
{genre= Western} => {cast1= William Boyd}	0.001203514	0.1041667	57.701389
{cast1= John Wayne} => {genre=Western}	0.001083163	0.8181818	51.895212
{crew1= Walt Disney} => {genre= Animation}	0.001564569	0.9285714	39.364796
{cast1= Clarence Nash} => {genre= Animation}	0.002768083	0.8518519	36.112434
{crew2= Walt Disney} => {genre= Animation}	0.002407029	0.8333333	35.327381
{crew3= Chiang Hsing-Lung} => {genre=Foreign}	0.001323866	1.0000000	25.724458
{crew1= Chang Cheh} => {genre=Foreign}	0.001444217	0.8571429	22.049536
{crew2= Ni Kuang} => {genre=Foreign}	0.001323866	0.7857143	20.212074
{crew1= Chuck Jones} => {genre=Family}	0.001203514	0.5882353	19.242705
{cast1= Mel Blanc} => {genre=Family}	0.001684920	0.4375000	14.311762
{cast1= Mel Blanc} => {genre= Animation}	0.001083163	0.2812500	11.922991
{crew1= Jules White} => {genre= Comedy}	0.001805271	0.9375000	10.470010
{cast1= Curly Howard} => {genre= Comedy}	0.001684920	0.9333333	10.423477
{cast1= Vijayakanth} => {genre= Drama}	0.001564569	0.7647059	8.449390
{crew1= Ilayaraja} => {genre= Drama}	0.002407029	0.6666667	7.366135

General Discussion on Association

We have generated a separate dataset (usualsuspects.csv) to perform association analysis on the usual casts in movies. The dataset includes for each actor the co-stars and crews who have worked with him/her in different movies. The apriori algorithm generates 49274 rules. The above table includes the top 20 non redundant rules. The apriori algorithm revealed a very satisfying "fringe case" in our relatively small data set, with a great deal of cultural relevance. These rules were dominated by cult filmmaker John Waters and various permutations of his recurring cast members including David Lochary, Mary Vivian Pearce, and of course the infamous persona known as Divine. For the uninitiated, John Waters is an eccentric filmmaker who happens to play a significant role in the identity of the city Baltimore, Maryland. He is known for writing, directing, and producing his own low-budget films (as reflected by his recurrence in multiple "crew" columns). Waters is also well known for using many of the same actors in his films. Because of his status as an "outsider artist", those involved in his films had careers in a sort of "fringe state", where many of his recurring cast members appeared almost exclusively in his films throughout their entire careers. The "outsider artist" status of Waters and company is reflected by the very high lift and low support seen in the corresponding rules.

We get only 11 meaningful rules which relate genre and cast in the original dataset. This is aligned with the results of classification because the low number of rules correspond to the low accuracy of the best classification algorithm on the dataset.

We get 17 meaningful rules after subsampling the original dataset, where we reduce the number of movies with genre as 'Others', creating an unbiased dataset. This is also aligned with the results of classification because the low number of rules correspond to the low accuracy of the best classification algorithm on the dataset.

Clustering

For clustering we used the ROCK Clustering algorithm. This was used as the clustering algorithm for this dataset after many of the traditional algorithms such as K-Means, DBSCAN could not work as the dataset contains both nominal and numeric data. The ROCK Clustering is present in the package "cba" and can be used by using the library "cba".

The "cba" package contains an inbuilt dataset called "Mushroom" which we have used in our clustering process in order to illustrate the difference in clustering based on the size of the dataset. The Mushroom dataset contains about 8000 data regarding the edibility of different mushrooms. This could be converted into a dummy dataset so as to reduce the computational time. The dataset which we have contains about 12000 data with various attributes and could not be converted into the dummy dataset and thus the final cluster graph is very condensed.

The rockCluster function was used to do the clustering. It first computes the link value for each set of points, i.e., transform the original similarities (computed by the Jaccard coefficient) into □similarities that reflect the number of shared neighbors between points. It then performs an agglomerative hierarchical clustering on the data using the "number of shared neighbors" similarities and the "maximize the shared neighbors" objective function and finally, assigns the remaining points to the clusters that have been found.

Here increasing the number of clusters to k' does not help in better categorization of data . The cluster values when increased give a more dense graph than the one which we have. So increasing the number of clusters does not help in the better categorization in this dataset.
Since the dataset contains both categorical and Boolean values the use of K-Means and DBSCAN was not possible and the usage of the "gower" metric in the Daisy Gower clustering could not be done due to the size of the dataset. So after running the various algorithms the ROCK clustering proved to be the most feasible and efficient one which could be used.

ROCK uses the following parameters for its execution:
X: a data matrix; for rockLink an object of class dist.
n: the number of desired clusters.
beta: optional distance threshold.
theta: neighborhood parameter in the range [0,1)

The output has the following parameters:

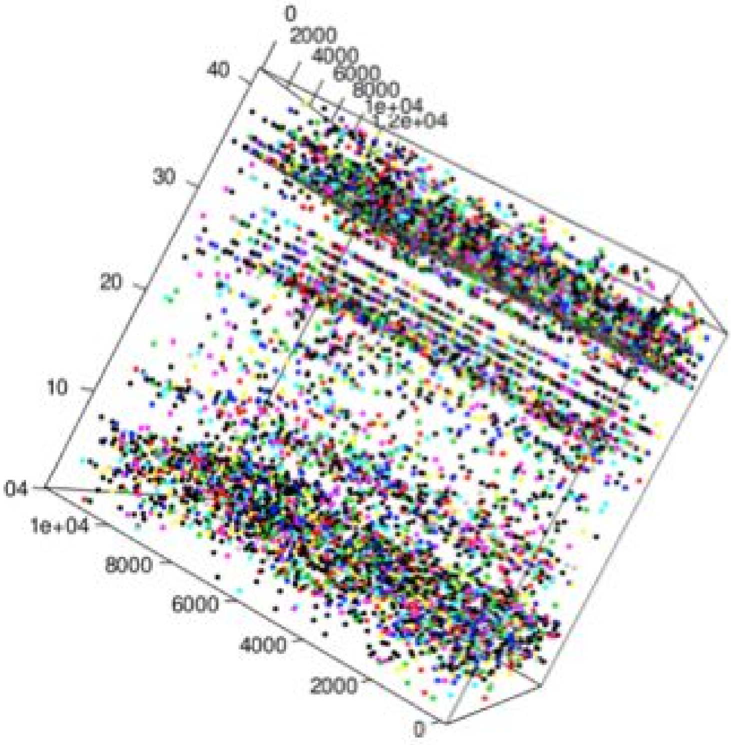
x: the data matrix or a subset of it.
cl: a factor of cluster labels.
size: a vector of cluster sizes.

The following diagram gives the final clustering graph of the algorithm for the dataset which was plotted with title and genre as the values plotted. Our output gave us 23 clusters after computing the distances and links.

```
Clustering:
computing distances ...
computing links ...
computing clusters ...
rockMerge: terminated with 23 clusters
```

After the clustering:

```
data: x
beta: 0.2
theta: 0.8
fun: dist
args: list(method = "binary")
1    2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
10  1  1  1  1  1  1  1  1  1   1   1   1   1   1   1   1   1   1   1   1   1   1
```



Conclusion

Through this project we have found some interesting information about the movie industry. As expected we can see through classification that there is indeed some relation between cast and crew members and the genres of movies that they feature in. For example you would expect to see Bruce Willis in primarily action movies. We have found through our association rules that there are some cast and crew members that prefer to work together. We can see that this is true in many real life scenarios, such as how Quentin Tarantino often hires several of the same actors and actresses for his movies. For clustering, the traditional clustering algorithms such as K-means and DBSCAN could not work since the features in the dataset are primarily categorical. In this case the ROCK clustering algorithm produced the best results giving us 23 clusters which is the same as the number of different genres in the dataset.

References

[1] http://cs229.stanford.edu/proj2008/KammHuangSathi-TheNetflixChallenge.pdf
[2] https://www.cis.upenn.edu/~sudipto/mypapers/categorical.pdf
[3] http://www.rdatamining.com/docs/association-rule-mining-with-r
[4] http://www.rdatamining.com/examples/association-rules
[5] http://www.r-bloggers.com/using-apply-sapply-lapply-in-r/
[6] www.stackoverflow.com
[7] https://cran.r-project.org/web/packages/arules/arules.pdf
[8] https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/RockCluster
[9] https://www.cis.upenn.edu/~sudipto/mypapers/categorical.pdf
[10] https://cran.r-project.org/web/packages/cba/cba.pdf
[11] https://en.wikipedia.org/wiki/Netflix_Prize
[12] http://kdd.org/exploration_files/6-Netflix-1.pdf
[13] https://www.igvita.com/2007/01/27/correlating-netflix-and-imdb-datasets/