Analysis on IMDB database

Group Members

Fatemeh Tavassoli - 18695225

Kingshuk Mukherjee - 92820942

Kunal Sonalkar – 76411926

Prerna Mandal - 29533906

Data Mining – Final Project

Fall 2015

Data mining on IMDb data – Project Report

1. Background on IMDb

The internet movie database is the largest database for movies on the internet. IMDb maintains an incredible amount of data on movies across the world. The data includes details on cast, production crew, writer, director, plot summary, reviews, etc. This data is made available to the public and can be used for various kinds of analysis and recommendation process.

As of November 2015, IMDb had approximately 3.5 million titles (includes episodes) and 6.8 million personalities in its database, as well as 64 million registered users and is an Alexa Top 50 site. The site allows users to rate movies add new titles and edit entries.

IMDb does not provide an API for automated queries. However, most of the data can be downloaded as compressed plain text files and the information can be extracted using the command-line interface tools provided. Beside that there is the Java-based graphical user interface (GUI) application available which is able to process the compressed plain text files and allow searching and displaying the information. A Python package called IMDbPY can also be used to process the compressed plain text files into a number of different SQL databases, which enables easier access to the entire dataset for searching or data mining

2. Literature review

Movie recommendation is one of the most popular fields in data mining and AI. Several websites recommend movies to users based on their tastes. Movie recommendation is such an interesting area in data mining because 1) The amount of available data on movies is immense and 2) There are several different mining approaches for the same problem. For good results, it is very important that the mining is done on the correct dataset using the proper approach/method.

The most popular web site for movie recommendation is perhaps IMDb itself. IMDb has a ton of information about movies and uses this information for recommendation. Let us look at how the IMDb choose personalized recommendations.First, ittakes all movies and TV shows that a user has either rated or added to hisWatchlist. Then, it compares this data to ratings made by other users. Using this comparison, they find movies and TV shows that people with similar tastes like and recommend those movies to the user.

Other popular movie recommendation engines include Netflix, Rotten Tomatoes, Movielens, Flixster, Crickter, Clerkdogs, Nanocrowd, Taste Kid and Jinni. They use a variety of different approaches and include different features. The semantic search on Jinni is worthy of special mention. Users can search for topics like ‘Movies on Boxing’ or ‘Movies about gangsters’ and the search returns good results.

There are a number of research papers on this topic as well. Most of the movie recommendation websites recommend movies based on the user’s data (of movies watched and liked). Less work has been done on using the features of the movie itself in the recommendation process. This has been done by Bo Li et al [1] using machine learning techniques to construct a cluster for the movie by implementing a distance matrix based on the movie features and then make movie recommendation in real time.

3. Dataset creation and summary

3.1 The problem

For the mining process it is very important to have a good dataset with the correct features. The IMDb contains a lot of data in the form of list files. We have to extract the relevant information from those files and parse them into a csv file which can readily be used in the mining process. There were two main challenges in this section. 1) Deciding which features to extract and 2) The actual process of data cleaning and parsing.

3.2 Features to be taken

After debating on which features to take, we decided to take all the features that appeared relevant in some way or the other. The idea was that by taking a superset of features at this point, we could then easily take a subset of these features for a specific mining process later on. Basically, we realized that for each mining operation, we would need different set of features. Therefore the best decision at this step was to take as many features as possible. For each mining operation, we have separately discussed which attributes which we have considered for that operation.

3.3 Summary of the extraction process

The dataset is obtained by combining several of list files provided by IMDB. These files include movies (which contain movie titles and years), genres, producers, directors, actors, running times and writers; each of which contain information about several hundred thousand movies.

Linux tools such as sed are used to align the text files, since they have different formats.Invalid characters were removed and extra descriptions in each file were deleted. There are also movie titles that have some extra information in front of them in the form of parentheses. These could be (VG), which indicates video game or some other information. These lines were ignored, because they are not movies.

Then the data is fed into a piece of Java code, prepared for this task, that loads the data from each file and then joins them on movie's title through hashing of movie titles. This is done for every movie title in genres file and if a movie is missing in genres.list or any of the other files; that movie is ignored. Each file contains widely varying number of movies; joining them on movie title leaves more than 140k movies with their year, genres, producers, directors, actors, writers and running times. It is possible that a movie has multiple actors and directors. The rows are repeated for genres, but for the other features, they are put into different fields, producer1, producer2, director1, director2, actor1, actor2, actor3, actor4, writer1 and writer2; and these fields could be repeated for the same movie title with multiple genres. Each movie has 1 to 5 genres.

We used 3 datasets primarily in this project

3.4 Dataset 1

For generating the dataset 1, we used the python package imdbpy.

To ensure we spanned across wide variety of major Hollywood movies we took data from http://www.imdb.com/list/ls057823854/ "**All Movies from 1972-2016**",

http://www.imdb.com/list/ls050597145/ "**Movies from 1950-1970**"

http://www.imdb.com/list/ls002502932/ "**Classics from 1940**".

Using variety of genres, actors, directors, writers, producers, etc helps us make sure that we are covering a wide range of movies.

Dataset summary:

The following table depicts the number of **unique** values present for each attribute considered.

Number of Titles	9940
Number of Writers	6705
Number of Directors	4100
Number of Producers	5804
Number of Casts	4336
Number of Languages	43
Number of Genres	23

Fields in the dataset:

MovieID, Title, Genre0, Genre1, Year, Cast0, Cast1, Director, Writer, Runtime, Kind, Distributor, Language, Country, Rating, Producer.

The following table shows the movie distribution of few prominent genres considered for analysis

Genre	Action	Comedy	Drama	Horror	Crime	Thriller
N.O. of Movies	1711	2617	2108	546	751	70
Genre	Biography	Adventure	Sci-Fi	Romance	Mystery	Western
N.O. of Movies	555	527	42	23	56	32

We use this dataset in our first attempt at classification (genre prediction). In this dataset, every movie has a single row. There are two fields, 'genre0' and 'genre1' which contain the first 2 genres for every movie.

3.5 Dataset 2

In this particular dataset we considered multiple genres for a particular movie. To be precise one movie had been assigned 2 rows if had 2 genres, 3 rows if it had 3 and so on so forth. One of the primary reasons for this was a movie can have more than one genre and hence this technique gave a significant improvement in classification as compared to first dataset where one movie had only one row assigned.

Cleaning and Data Processing for Dataset 2

There are several side-tasks that were done to clean the dataset further using Python scripting:

From all the rows, there were only ~1000 lines with invalid or weird genres, which were removed; these weird names included "**?"** or **"reality-tv"** etc. which weren't part of movies.

These are the genres that are kept:

"Action","Adult","Adventure","Animation",

"Biography","Comedy","Crime","Documentary","Drama",

"Family","Fantasy","History","Horror","Music","Musical","Mystery",

"Romance","Sci-Fi","Short","Sport","Thriller","War","Western".

Also, only ~6k out of more than 140k movies has second directors, so these second directors are being removed.

Key Features:

- Any movie that had anything besides **a-z, A-Z or space** in the title were completely removed, this still leaves more than 100k movies to choose from.
- Removing movies before 1960 leaves around 78k movies; this was done to do the analysis on more recent movies.
- Number of movies per director was also counted, and the top 200 were chosen, but it would leave only ~6000 movies, so the **data was sorted on directors' movie count** and any **director that has over 5 movies were kept**, this leaves more than 16k movies; which is desirable.

Dataset summary:

Number of entries: 16,567, Number of Titles: 7335, Number of Genres: 23

3.6 Dataset 3

Dataset 3 generation has been explained under the section titled **"A better Strategy"** which uses the **"Bag of Words"** method for genre prediction.

4. Genre prediction

4.1 The problem

For the genre prediction, we have to take a set of movies with known genres and train a model on their features with genre as the class attribute. After training the model on a training set, we have to run it on the test set, and predict the genre of the movies in the test set.

This is basically a classification problem. There are a few different ways to approach this problem. The methods that we have covered in class for classification are Decision tree, Ripper, C4.5, Oblique decision trees and Naïve Bayes classifier. Each method has its own advantages and is useful for certain problems.

4.2 Why Naïve Bayes?

For this problem, we have used Naïve Bayes as our classification model. Naïve Bayes predicts the class by calculating the posterior probability of each class based on the attributes of the test data. The predicted class is the one with the highest probability. In our problem, the attributes take a lot of different alternatives (values) and are independent. For example the attribute director takes as many different values as the number of different directors in the dataset. Also the different attributes like the director, the writer, the producer etc can be considered to be independent attributes. The size of our dataset is also significantly big which means that most classification methods would not be suitable.

Conceptually the way we saw the classification model should work is this:

During training, each variable should be associated with a genre with certain probability depending on the relative number of times that variable takes part in a movie of that genre.

During testing, the variables of the test instance should be considered and depending on the combined probability of the variables, a prediction of the genre can be done (the genre with the highest probability for that combination of variables should be predicted).

For example, say in the test instance, we have Director= X and Writer=Y. From the training model, we already know the probability with which variables X and Y are involved in each genre. Considering them as independent variables, we find out the joint probability of X and Y being involved in each genre. Then we predict the genre which has the highest probability of involvement.

Naïve Bayes classifier works exactly in this manner by calculating the posterior probabilities of different classes. Since it matched with our conceptual understanding of the problem, we decided to use Naïve Bayes classifier.

4.3 Dataset (Attributes used)

Next comes the question: what data is relevant for this classification problem? The dataset that we created in the last step (dataset1) contains the following attributes: Movie title, year of release, genre, director, writer, producer, cast 1, cast 2, running time, distributers etc. Taking all these features for training the model will not be a smart idea. We should take only those attributes which have some impact on the genre- not random attributes which have little or no effect. For example, the attributes like movie title, year of release, running time, distributers- have little or no impact on the genre.

Intuitively we can tell that the attributes that can have impact on the genre are: director, writer, cast and producer. Among these the director and writer appear to be the ones which are most significant. A director mostly directs certain 'type' of movies and a writer generally writes certain 'type' of stories. Here 'type' represents the genre. Using this intuition, we have trained the model. Later on, we have included a 3rd attribute and shown that the accuracy does not improve thereby proving our hypothesis.

4.4 Implementation and results

We implemented the Naïve bayes classification on R.

The dataset had the following features: Genre, Director, and Writer with genre being the class attribute.

Here we faced a problem because many movies had multiple genres. Infact, most of the prominent movies have 2-3 genres. Our first approach was to just take the first genre (genre0) for every movie and train the classifier on that.

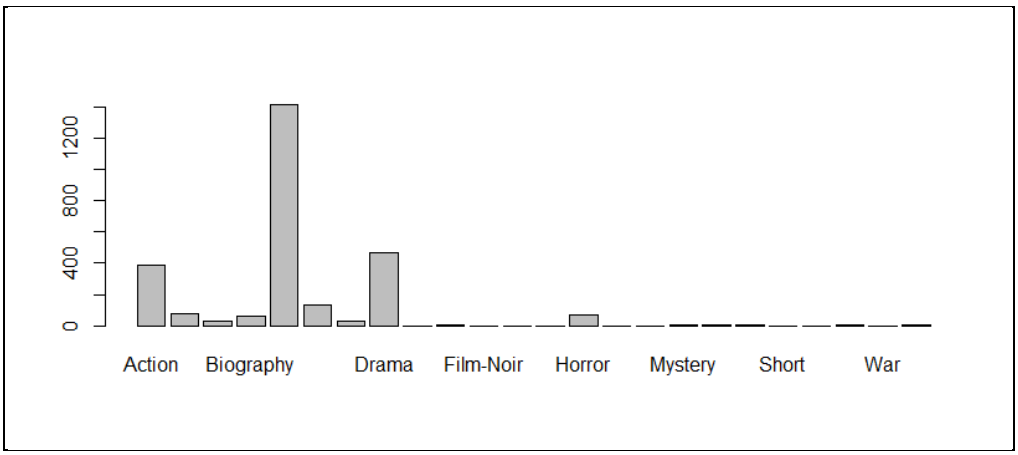
	genre0	director	writer
1	Action	Marc Webb	Alex Kurtzman
2	Crime	Frank Darabont	Stephen King
3	Action	George Lucas	George Lucas
4	Adventure	Robert Zemeckis	Robert Zemeckis
5	Comedy	John Hughes	John Hughes
6	Adventure	Richard Donner	Steven Spielberg
7	Crime	Jonathan Demme	Thomas Harris
8	Adventure	Steven Spielberg	Michael Crichton
9	Animation	Roger Allers	Irene Mecchi
10	Crime	David Fincher	Andrew Kevin Walker

Using this approach, we got a low accuracy of **33.15%**

```
Overall Statistics

Accuracy : 0.3315
                 95% CI : (0.3136, 0.3497)
    No Information Rate : 0.2714
    P-Value [Acc> NIR] : 4.144e-12

Kappa : 0.1526
Mcnemar's Test P-Value : NA
```



Predictions plot

We changed the attributes being considered and took into account **(cast0, writer, director, genre1)**. We got accuracy around **47%**

```
Overall Statistics

               Accuracy: 0.4701
                 95% CI: (0.4473, 0.493)
    No Information Rate: 0.2593
    P-Value [Acc> NIR]:< 2.2e-16
```

```
                  Kappa: 0.3727
Mcnemar's Test P-Value: NA
```

.4.5 Improving the classification

It was clear that by using the approach of considering only 1 genre for training, we could not get a better accuracy. Also, it was an inaccurate training model in the first place as the movies with multiple genres had elements of different genres and by taking only the first genre we were basically limiting the classification model.

Our next approach was to consider all the genres for every movie. For implementing this, we modified our dataset to contain multiple rows for movies with multiple genres with each row representing one genre.

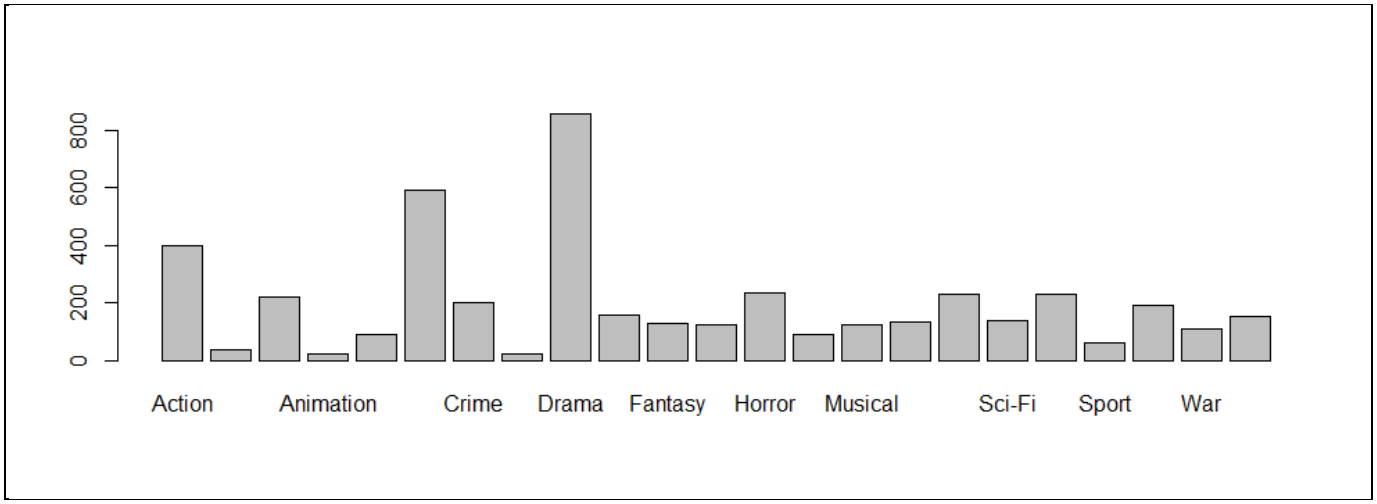
	title	genre	director1	writer1
1	Apeun seongsuk	Drama	Park Cheol-su	Kang Dae-ha
2	Apeun seongsuk	Romance	Park Cheol-su	Kang Dae-ha
3	Vigilante Force	Action	Armitage George	Armitage George
4	Vigilante Force	Adventure	Armitage George	Armitage George
5	Vigilante Force	Crime	Armitage George	Armitage George
6	Vigilante Force	Drama	Armitage George	Armitage George
7	Vigilante Force	Thriller	Armitage George	Armitage George

After training the model on this dataset and testing it on the test set, we found that the accuracy has dropped to **18.43%**

```
Overall Statistics

Accuracy : 0.1843
                 95% CI : (0.1732, 0.1959)
    No Information Rate : 0.2255
    P-Value [Acc> NIR] : 1

Kappa : 0.1028
Mcnemar's Test P-Value : NA
```



This was an unexpected result at first. Then after some analysis, we realized the reason why we were getting such a low accuracy was because each movie had multiple entries for multiple genres and only one (or more) of them can be in the (randomly divided) test set. While predicting, the Naïve bayes just predicts the most likely genre for every movie. That predicted genre has a low probability of matching with the same genre as in the test set. So even if the predicted genre matches with one of the actual genres of the movie, if it does not match with the genre of the entry in the test set, its accuracy would be reported as zero by the confusion matrix.

For example, say we have 5 movies, each with 3 genres. In our test set, we have, say 5 entries- one for each movie. Now let us assume that our classifier predicts one correct genre for every movie. In this case the accuracy should be 100%. But if, say, all the entries in the test set has a different genre than the one predicted by the classifier, the confusion matrix would report the accuracy as 0.

To overcome this hurdle, we wrote a method that would allow us to calculate the real accuracy for our prediction. The method takes the prediction for every title in the test set, goes back to the complete dataset and finds out all the real genres of that title and then matches the predicted genre with the real genres. If a match is found then it is a HIT.

```
[1] Comedy
[1] Family  Drama   Musical Romance

[1] Action
[1] Thriller Drama

[1] Western
[1] "HIT!"
[1] Adventure Thriller  Western   Drama

[1] Comedy
[1] "HIT!"
[1] Comedy   Mystery  Crime    Thriller

[1] Musical
[1] "HIT!"
[1] Musical Family  Drama

[1] Drama
[1] Comedy

[1] Drama
[1] "HIT!"
[1] Drama     Action    Thriller  Adventure

[1] Drama
[1] Action Crime
```

```
[1] Comedy
[1] Drama   Romance

[1] Comedy
[1] "HIT!"
[1] Comedy  Romance Crime

[1] Drama
[1] Sci-Fi    Adventure Action

[1] Short
[1] "HIT!"
[1] Short    Drama    Thriller Mystery
Romance

[1] Short
[1] "HIT!"
[1] Comedy Short

[1] Adventure
[1] "HIT!"
[1] Action    Adventure Sci-Fi    Thriller

[1] Horror
[1] Thriller Action   Crime
```

This is a working example of the method for 15 entries in the test set. The first line represents the predicted genre and the next line represents the actual genres of that title. If one of the actual genres matches with the predicted genre then a HIT is reported and it is counted as a successful prediction.

The accuracy reported by this method is **68.41%**

```
print(accuracy)
[1] 0.6841068
```

During different runs of this method with different (randomized) datasets, we got accuracy in the range of 66% to 71%. This in our opinion represents the best accuracy that can be achieved using this classification technique on this dataset.

4.6 Including a 3rd attribute

As discussed in section 4.3, we performed our classification on features director and writer. In this section we would include producer and cast1 as 3rd attributes and find out if they improve the accuracy.

With producer as the 3rd attribute:

	title	genre	director1	producer1	writer1
1	Apeun seongsuk	Drama	Park Cheol-su	Choi Chun-ji	Kang Dae-ha
2	Apeun seongsuk	Romance	Park Cheol-su	Choi Chun-ji	Kang Dae-ha
3	Vigilante Force	Action	Armitage George	Corman Gene	Armitage George
4	Vigilante Force	Adventure	Armitage George	Corman Gene	Armitage George
5	Vigilante Force	Crime	Armitage George	Corman Gene	Armitage George
6	Vigilante Force	Drama	Armitage George	Corman Gene	Armitage George
7	Vigilante Force	Thriller	Armitage George	Corman Gene	Armitage George

Accuracy= **67.03%**

With actor1 as the 3rd attribute:

	title	genre	director1	writer1	actor1
1	Apeun seongsuk	Drama	Park Cheol-su	Kang Dae-ha	Yu In-chon
2	Apeun seongsuk	Romance	Park Cheol-su	Kang Dae-ha	Yu In-chon
3	Vigilante Force	Action	Armitage George	Armitage George	Kristofferson Kris (I)
4	Vigilante Force	Adventure	Armitage George	Armitage George	Kristofferson Kris (I)
5	Vigilante Force	Crime	Armitage George	Armitage George	Kristofferson Kris (I)
6	Vigilante Force	Drama	Armitage George	Armitage George	Kristofferson Kris (I)
7	Vigilante Force	Thriller	Armitage George	Armitage George	Kristofferson Kris (I)

Accuracy= **68.16%**

As we can see, adding a 3rd attribute does not improve the accuracy of classification. Using the concept of Occam's Razor- Given other things remain same, the least complex model is the one we should take-, we can, therefore, conclude that using director and writer gives us the best model.

5. Association rules

5.1 The problem

For this problem, we need to find the list of usual casts- the people who usually work together. In every movie we have the cast members, production crew, writer, director, and producer. We need to find the list of top 20 most frequent casts.

This is basically a problem where we have to find association rules between the fields. The association rule miming techniques covered in class are Apriori technique and FP growth tree. For this problem, the Apriori technique is very well suited.

5.2 Dataset (attributes used)

For the association rule mining, we need to set up the dataset in such a way so that there is one row for every movie instance and it includes the features: cast0, cast1, director, writer and producer.

	cast0	cast1	director	writer	producer
1	Andrew Garfield	Emma Stone	Marc Webb	Alex Kurtzman	Avi Arad
2	Tim Robbins	Morgan Freeman	Frank Darabont	Stephen King	Liz Glotzer
3	Mark Hamill	Harrison Ford	George Lucas	George Lucas	Gary Kurtz
4	Michael J. Fox	Christopher Lloyd	Robert Zemeckis	Robert Zemeckis	Neil Canton
5	Emilio Estevez	Paul Gleason	John Hughes	John Hughes	Gil Friesen
6	Sean Astin	Josh Brolin	Richard Donner	Steven Spielberg	Harvey Bernhard
7	Jodie Foster	Lawrence A. Bonney	Jonathan Demme	Thomas Harris	Grace Blake
8	Sam Neill	Laura Dern	Steven Spielberg	Michael Crichton	Kathleen Kennedy
9	Matthew Broderick	Jonathan Taylor Thomas	Roger Allers	Irene Mecchi	Alice Dewey
10	Morgan Freeman	Andrew Kevin Walker	David Fincher	Andrew Kevin Walker	Stephen Brown
11	Tom Hanks	Tim Allen	John Lasseter	John Lasseter	Bonnie Arnold
12	Jeff Bridges	John Goodman	Joel Coen	Ethan Coen	Tim Bevan
13	Tom Hanks	Tom Sizemore	Steven Spielberg	Robert Rodat	Ian Bryce
14	Edward Norton	Brad Pitt	David Fincher	Chuck Palahniuk	Ross Grayson Bell
15	Keanu Reeves	Laurence Fishburne	Andy Wachowski	Andy Wachowski	Bruce Berman
16	Liam Neeson	Ewan McGregor	George Lucas	George Lucas	George Lucas
17	Russell Crowe	Joaquin Phoenix	Ridley Scott	David Franzoni	David Franzoni
18	Hugh Jackman	Patrick Stewart	Bryan Singer	Tom DeSanto	Avi Arad
19	Richard Harris	Maggie Smith	Chris Columbus	J.K. Rowling	Todd Arnow
20	Alan Howard	Noel Appleby	Peter Jackson	J.R.R. Tolkien	Peter Jackson
21	Bruce Allpress	Sean Astin	Peter Jackson	J.R.R. Tolkien	Peter Jackson
22	Vin Diesel	Asia Argento	Rob Cohen	Rich Wilkes	Creighton Bellinger
23	Albert Brooks	Ellen DeGeneres	Andrew Stanton	Andrew Stanton	Jinko Gotoh
24	Noel Appleby	Ali Astin	Peter Jackson	J.R.R. Tolkien	Peter Jackson
25	Will Ferrell	Christina Applegate	Adam McKay	Will Ferrell	Judd Apatow

Additional features such as cast2, cast3, assistant director, assisting writers and additional producers can be also added but since they are not significant in most movies, we have decided to work with these 5 features.

5.3 Implementation and results

We have implemented the apriori algorithm on R.

We have only used the parameters MinLength, MinSupport and MinConfidence and not other parameters which can be used to get rules for certain RHS. The reason behind this is we want the apriori algorithm to first find us the most significant association rules irrespective of who appears on the RHS.

To start with, we took **MinLength= 2, MinSupport= 0.008, MinConfidence=0.5**

After getting rid of the redundant rules and sorting the significant rules according to the lift, we get the following 20 most significant rules.

MinLength=2/MinSupport= 0.0008/MinConfidence= 0.5

	lhs		rhs	support	confidence	lift
1	{producer=Jules Bass}	=>	{director=Jules Bass}	0.0008238929	1.0000000	1213.7500
2	{director=Jim Jarmusch}	=>	{writer=Jim Jarmusch}	0.0008238929	1.0000000	1213.7500
3	{producer=John Boorman}	=>	{director=John Boorman}	0.0008238929	1.0000000	1078.8889
4	{director=Mike Leigh}	=>	{writer=Mike Leigh}	0.0009268795	1.0000000	1078.8889
5	{director=Quentin Tarantino}	=>	{writer=Quentin Tarantino}	0.0008238929	1.0000000	971.0000
6	{writer=Peter Greenaway}	=>	{director=Peter Greenaway}	0.0009268795	1.0000000	882.7273
7	{writer=M. Night Shyamalan}	=>	{director=M. Night Shyamalan}	0.0010298661	1.0000000	882.7273
8	{director=Richard Schickel}	=>	{writer=Richard Schickel}	0.0009268795	0.9000000	873.9000
9	{writer=Robert Rodriguez}	=>	{director=Robert Rodriguez}	0.0008238929	0.8888889	863.1111
10	{cast0=ToshirÃ´ Mifune}	=>	{director=Akira Kurosawa}	0.0009268795	1.0000000	809.1667
11	{director=Ingmar Bergman}	=>	{writer=Ingmar Bergman}	0.0010298661	0.9090909	802.4793
12	{director=Mike Leigh}	=>	{producer=Simon Channing Williams}	0.0008238929	0.8888889	784.6465
13	{writer=Mike Leigh}	=>	{producer=Simon Channing Williams}	0.0008238929	0.8888889	784.6465
14	{cast0=Tyler Perry, writer=Tyler Perry}	=>	{director=Tyler Perry}	0.0009268795	1.0000000	746.9231
15	{writer=Tyler Perry}	=>	{director=Tyler Perry}	0.0013388260	0.9285714	693.5714
16	{producer=Malcolm Craddock}	=>	{cast1=Daragh O'Malley}	0.0008238929	1.0000000	693.5714
17	{writer=Werner Herzog}	=>	{director=Werner Herzog}	0.0011328527	1.0000000	693.5714
18	{cast0=Tyler Perry}	=>	{director=Tyler Perry}	0.0009268795	0.9000000	672.2308
19	{director=Robert Rodriguez}	=>	{producer=Elizabeth Avellan}	0.0008238929	0.8000000	647.3333
20	{director=James Cameron}	=>	{writer=James Cameron}	0.0008238929	0.8000000	647.3333

With minlength=2 we get interesting results. It is easy to identify artists who play double roles in their movies. The most common combination is writer+director and we can find many popular and critically acclaimed directors in our top 20 list.

However, with Minlength=2, we donot get many rules for longer association in the top 20 list (only rule we get is #14 in which Tyler Perry plays the cast0, director and writer). This is expected because the support for length=2 is greater than the support for length=3. If we want to check for larger association rules to identify people who work together, we have to test it for Minlength=3. We also need to lower our MinSupport for getting longer associations.

So, next we find association rules for MinLength=3, MinSupport= 0.0005, MinConfidence=0.5

After getting rid of the redundant rules and sorting the significant rules according to the lift, we get the following 20 most significant rules.

Minlength= 3/ MinSupport= 0.0005/ MinConfidence= 0.5

	lhs	rhs	support	confidence	lift
1	{director=Paul Thomas Anderson, writer=Paul Thomas Anderson}	=> {producer=Paul Thomas Anderson}	0.0005149331	0.8333333	1618.3333
2	{cast0=Bruce Balden, cast1=Jacqueline Bassett}	=> {producer=Michael Apted}	0.0006179197	1.0000000	1618.3333
3	{cast0=Bruce Balden, director=Michael Apted}	=> {cast1=Jacqueline Bassett}	0.0006179197	1.0000000	1618.3333
4	{cast0=Bruce Balden, director=Michael Apted}	=> {producer=Michael Apted}	0.0006179197	1.0000000	1618.3333
5	{cast1=Jacqueline Bassett, director=Michael Apted}	=> {producer=Michael Apted}	0.0006179197	1.0000000	1618.3333
6	{director=Rocco Urbisci, writer=George Carlin}	=> {cast0=George Carlin}	0.0005149331	1.0000000	1618.3333
7	{cast1=Robert Lindsay, producer=Andrew Benson}	=> {director=Andrew Grieve}	0.0006179197	1.0000000	1618.3333
8	{cast0=Ioan Gruffudd, cast1=Robert Lindsay}	=> {director=Andrew Grieve}	0.0006179197	1.0000000	1618.3333
9	{cast0=Ioan Gruffudd, producer=Andrew Benson}	=> {cast1=Robert Lindsay}	0.0006179197	1.0000000	1618.3333
10	{cast0=Ioan Gruffudd, producer=Andrew Benson}	=> {director=Andrew Grieve}	0.0006179197	1.0000000	1618.3333
11	{director=Mel Brooks, writer=Mel Brooks}	=> {cast0=Mel Brooks}	0.0005149331	0.7142857	1387.1429
12	{writer=Dario Argento, producer=Claudio Argento}	=> {director=Dario Argento}	0.0005149331	1.0000000	1387.1429
13	{cast0=William Shatner, writer=Gene Roddenberry}	=> {cast1=Leonard Nimoy}	0.0005149331	1.0000000	1387.1429
14	{cast1=Daragh O'Malley, writer=Bernard Cornwell}	=> {producer=Malcolm Craddock}	0.0006179197	1.0000000	1213.7500
15	{director=Tom Clegg, writer=Bernard Cornwell}	=> {producer=Malcolm Craddock}	0.0006179197	1.0000000	1213.7500
16	{cast0=Sean Bean, writer=Bernard Cornwell}	=> {producer=Malcolm Craddock}	0.0006179197	1.0000000	1213.7500
17	{writer=Quentin Tarantino, producer=Lawrence Bender}	=> {director=Quentin Tarantino}	0.0005149331	1.0000000	1213.7500
18	{writer=Mel Brooks, producer=Mel Brooks}	=> {director=Mel Brooks}	0.0005149331	0.8333333	1155.9524
19	{director=Mike Leigh, producer=Simon Channing Williams}	=> {writer=Mike Leigh}	0.0008238929	1.0000000	1078.8889
20	{director=Ingmar Bergman, writer=Ingmar Bergman}	=> {producer=Allan Ekelund}	0.0005149331	0.5000000	971.0000

Going over these rules we can identify groups of people who work together more often. The top 20 results still includes few people playing multiple roles in the same movie. The reason behind this is there are certain people who play multiple roles in all or most movies they make.

Next we try to find association rules for even longer association of MinLength=4

Minlength=4/ MinSupport= 0.0005/ MinConfidence= 0.5

	lhs	rhs	support	confidence	lift
1	{cast0=Bruce Balden, cast1=Jacqueline Bassett, director=Michael Apted}	=> {producer=Michael Apted}	0.0006179197	1.0000000	1618.3333
2	{cast0=Ioan Gruffudd, cast1=Robert Lindsay, producer=Andrew Benson}	=> {director=Andrew Grieve}	0.0006179197	1.0000000	1618.3333
3	{cast1=Daragh O'Malley, director=Tom Clegg, writer=Bernard Cornwell}	=> {producer=Malcolm Craddock}	0.0006179197	1.0000000	1213.7500
4	{cast0=Sean Bean, cast1=Daragh O'Malley, writer=Bernard Cornwell}	=> {producer=Malcolm Craddock}	0.0006179197	1.0000000	1213.7500
5	{cast0=Sean Bean, director=Tom Clegg, writer=Bernard Cornwell}	=> {producer=Malcolm Craddock}	0.0006179197	1.0000000	1213.7500
6	{cast0=Tyler Perry, director=Tyler Perry, writer=Tyler Perry}	=> {producer=Roger M. Bobb}	0.0005149331	0.5555556	899.0741
7	{cast0=Sean Bean, director=Tom Clegg, writer=Bernard Cornwell}	=> {cast1=Daragh O'Malley}	0.0006179197	1.0000000	693.5714
8	{cast0=Sean Bean, director=Tom Clegg, producer=Malcolm Craddock}	=> {cast1=Daragh O'Malley}	0.0008238929	1.0000000	693.5714
9	{cast1=Mia Farrow, director=Woody Allen, writer=Woody Allen}	=> {producer=Robert Greenhut}	0.0005149331	1.0000000	462.3810
10	{cast0=Woody Allen, director=Woody Allen, producer=Robert Greenhut}	=> {writer=Woody Allen}	0.0005149331	1.0000000	269.7222

This gives us association rules which have 4 characters.

5.4 Scope for Improvement

For this part, we cannot get drastically improved rules with the same dataset. If we have more movie entries in the dataset, it may give us some more interesting association rules. By taking in more features such as additional producers, writers, directors and an extended cast list, we can get some interesting results.

Basically for this problem, it all comes down to the dataset. In our opinion, the results that we have got with our dataset are reasonably good and interesting.

6. Clustering

6.1 The problem

Performing clustering on such a huge dataset was an attempt from our side to group similar movies together. This grouping was performed by considering factors such as **(Cast0, Cast1, Writer, Director, Producer and Genre).**

We considered 'K' clusters for this purpose as it indicated the number of genres present in our dataset. So we have tried to cluster the movies according to their genre in the best possible way.

6.2 Our approach

We have used K-modes approach to cluster the movies of IMDb database. We chose this method as K-means algorithm only works on numerical data because of the objective function it minimizes. A simple way to cluster nominal/categorical data is by using K-modes algorithm, which is similar to K-means in principle but uses modes instead of means and thus the objective function is different than K-means. Likewise K-means, the K-modes algorithm is O(N) and is more scalable than distance matrix based / hierarchical clustering because they are O(N^2), where N is the number of data objects.

We also tried other functions like Daisy (package: Cluster) which achieves handling of nominal and ordinal data by using the dissimilarity coefficient of "Gower". This coefficient is used as a metric when we encounter nominal data. It basically computes all the pair wise dissimilarities between the observations in the dataset and variables can be of mixed type. But this process was computationally very expensive as the size of the considered dataset was huge and the accuracy was also not very satisfactory.

6.3 Implementation and results

For implementation K-modes method we used package called "klar". We used the following attributes:

	genre0	cast0	cast1	director	writer	producer
1	Action	Andrew Garfield	Emma Stone	Marc Webb	Alex Kurtzman	Avi Arad
2	Crime	Tim Robbins	Morgan Freeman	Frank Darabont	Stephen King	Liz Glotzer
3	Action	Mark Hamill	Harrison Ford	George Lucas	George Lucas	Gary Kurtz
4	Adventure	Michael J. Fox	Christopher Lloyd	Robert Zemeckis	Robert Zemeckis	Neil Canton
5	Comedy	Emilio Estevez	Paul Gleason	John Hughes	John Hughes	Gil Friesen
6	Adventure	Sean Astin	Josh Brolin	Richard Donner	Steven Spielberg	Harvey Bernhard
7	Crime	Jodie Foster	Lawrence A. Bonney	Jonathan Demme	Thomas Harris	Grace Blake
8	Adventure	Sam Neill	Laura Dern	Steven Spielberg	Michael Crichton	Kathleen Kennedy
9	Animation	Matthew Broderick	Jonathan Taylor Thomas	Roger Allers	Irene Mecchi	Alice Dewey
10	Crime	Morgan Freeman	Andrew Kevin Walker	David Fincher	Andrew Kevin Walker	Stephen Brown
11	Animation	Tom Hanks	Tim Allen	John Lasseter	John Lasseter	Bonnie Arnold

We tried analysis with combinations of other attributes like Runtime, Distributors and Rating but the above figure shows the attributes which gave the best results. We also set the **number of clusters = number of genres** in the dataset to get the most optimum results.

The following table shows the distribution of the movies based on the genre.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Action	19	1601	1	0	0	0	0	0	0	0	33	0	1	0	0	0	0	0	0	0
Adventure	4	2	500	0	0	0	7	0	3	0	0	0	0	0	0	0	0	1	0	0
Animation	0	1	2	0	1	0	0	0	0	0	1	298	0	0	0	0	0	0	0	0
Biography	3	0	3	531	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0
Comedy	8	4	4	31	2477	0	5	0	0	1	0	0	0	20	4	0	0	13	0	0
Crime	729	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	11	0	0	4
Documentary	175	0	1	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Drama	14	2	8	26	18	1987	0	7	0	0	0	0	0	0	1	0	0	1	6	0
Family	46	0	0	0	0	2	2	0	0	0	0	2	0	0	0	0	0	0	1	0
Fantasy	42	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Film-Noir	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
History	11	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Horror	18	0	0	2	3	0	0	1	1	0	0	0	0	4	0	3	1	0	0	0
Music	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Musical	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mystery	52	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Romance	20	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sci-Fi	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Short	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sport	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Thriller	61	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2
War	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Western	27	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Cluster Validation

Genre	Movies in Cluster	Actual N.O. of Movies	Accuracy
Action	1566	1662	94.2%
Adventure	509	520	97.8%
Crime	721	747	96.5%
Comedy	2376	2594	91%
Animation	320	325	98.4%
Drama	2006	2077	96.5%
Mystery	51	56	91%
Horror	518	538	96.28%
Western	24	32	75%
Thriller	63	66	95.45%
Biography	538	549	97.9%
Sci-Fi	37	41	90.2%
Fantasy	1	48	0.02%
Family	2	53	3%
Documentary	1	199	0.5%
Romance	1	22	4.5%
Music	0	4	0%
War	1	4	25%

6.4 Scope for Improvement

There are other techniques and functions like “Daisy” and “Agnes” but we get more or less the same accuracy in these techniques too.

For getting better results we can increase the number of clusters. As of now the number of clusters has been set to the number of genres in the dataset, but if we increase the number of clusters by 3-4, we certainly get results but as the size of the dataset is huge it becomes extremely computationally expensive.

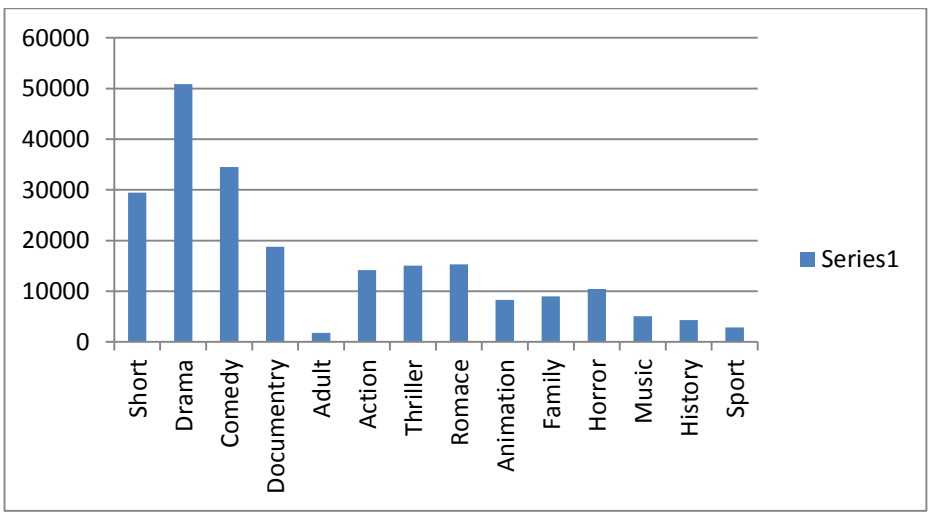
7. A better strategy

7.1 Introduction

After achieving not so good results with the normal approach discussed before, a better strategy came to mind. The new approach is based on how humans might be able to infer genre of a movie. Companies, distributers, directors or actors are not good indicators of movie genre; there are too many actors who play widely varying roles and few companies producing a lot of different movies. But one would be able to guess the genre of a movie with great accuracy by reading the movie plots' summary. This approach is a lot more robust than the previous one. The previous approach was not only giving poor results, but it was also limited to the directors and actors contained in the training set; a new movie with a new cast could not possibly be correctly classified. But plot summaries are always available for movies before they are even released; and the same set of English words are used to create them. This enables the second approach to be robust in face of new data points, ones never seen before; thus it generalizes better. It's also more robust to noises in the training data, directors or actors in movies with different genres are not important anymore. This approach is based on Natural Language Processing (NLP), uses bag of words technique, and we'll soon find out that the accuracy of this method is also immensely better than anything achieved before.

7.2 Dataset Creation

To able to work with plot summary, which is in plain text; one would have to first transform the data into a format that could be easily loaded and processed. The files used in this stage are movies.list, genres.list, keywords.list and plots.list. These files are merged on movie title so that for every title its genre and plot summary is present. This part was done using a combination of *nix sed program (for some stream editing and alignment of files) and a piece of Java code to merge the files using hash of movie titles. The next step is to transform the target function into a format that can be used for classification. Some movies have multiple genres and the data is suitable for one-versus-all classification and since a lot of classification functions work well and fast on binary targets, the training set is divided into several fields. The first field is the movie title. The second field is a string containing the whole plot summary of the movie. The other fields are binary, denoting the genres the movie has. For example, a movie could have it's 'Drama' dimension as 0, 'Action' as 1, 'Thriller' as 1, 'Romance' as 0 and so on. This type of feature generation would make it possible to predict multiple genres for a single movie as the program is able to answer queries in the form of "Is this movie a 'Drama'?", "Is this movie a 'Romance'?", etc. and the answer for several of them could potentially be 'yes'.



	title	keywords	Short	Drama	Comedy	Documentary	Adult	Action	Thriller	Romance	Animation	Family	Horror	Music	History	Sport
0	Paper Mountain (2005)	cornwall fair-trade maoist nepal In the Kingd...	1	0	0	1	0	0	0	0	0	0	0	0	0	0
1	The Hermits of Silicon Valley (2008)	independent-film satire urban-setting In this...	1	1	1	0	0	1	0	1	0	0	0	0	0	0
2	Lords of Dogtown (2005)	1970s accident alcoholic-mother auto-theft bar...	0	1	0	0	0	0	0	0	0	0	0	0	0	1
3	86000 Seconds: Sometime Someday in America (2002)	african-american american-dream limousine los-...	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Total movies: 124130

Total genres: 219 901 (one movie can have more than one genre)

7.3 Feature generation

Now that the data is in a desirable format, the next step is running the plot summary per movie through a NLP pipeline and priming it for classification. This stage is carried out in Python; using a combination of powerful libraries, frameworks and toolkits including numpy, pandas, re, nltk and scikit. The steps include loading the data using pandas. Then a function is defined that does several preprocessing steps:

1. Removal of obscure characters, limiting the plots to alphabetic words only that are only made up of a-z and/or A-Z.
2. Converting all upper-case letters to lower case to prevent differences for the same words.
3. Splitting the plot into a list of separate words.
4. Removal of stop-words (Optional step, since use of n-grams could indeed produce better results than the basic 1-gram model and n-gram models work better and are able to better capture the semantics of sentences if stop-words are not removed)
5. Stemming of the words from previous step. For this step, porter stemmer is used, which is available in NTLK package. This is done to prevent words such as ‘go’ and ‘going’ from becoming different features; while semantically they’re the same.
6. Lemmatizing the words from the previous step. This is necessary to remove variant forms of the same words, for example ‘wolf’ and ‘wolves’ should be the same.

This function is called for each movie’s plot summary and then kept in a list for every movie. This list is then fed into CountVectorizer to produce the bag of words. All words that appear in more than 95% of plot summaries are removed, n-grams of size 1 up to 4 are all generated and up to 40000 features are kept. Then using NLTK’s functions, the model learns the vocabulary and then transforms the training set which was a list of strings, one for every movie, into feature vectors. Each movie is now represented as a vector of 40k features, with each features value being equal to the number of times that feature (word) appears in the plot summary and 0 if it doesn’t appear. Another step to further improve the results is use of TF-IDF. TF-IDF, which is short for term frequency–inverse document frequency, is a numerical statistical method that is used to reflect how important a word is to a document in a collection or corpus. TF-IDF is applied to feature

vectors to give different weights to different features. The resulting data set is of shape (X, 40000) with X being the number of movie titles; which is kept in a compressed sparse matrix.

7.4 Example Run on Godfather Movie plot

The Godfather "Don" Vito Corleone is the head of the Corleone mafia family in New York. He is at the event of his daughter's wedding. Micheal, Vito's youngest son and a decorated WW II Marine is also present at the wedding. Micheal seems to be uninterested in being a part of the family business. Vito is a powerful man, and is kind to all those who give him respect but is ruthless against those who do not. But when a powerful and treacherous rival wants to sell drugs and needs the Don's influence for the same, Vito refuses to do it. What follows is a clash between Vito's fading old values and the new ways which may cause Micheal to do the thing he was most reluctant in doing and wage a mob war against all the other mafia families which could tear the Corleone family apart.

Step1:Removing [^a-zA-Z]

The Godfather Don Vito Corleone is the head of the Corleone mafia family in New York He is at the event of his daughter s wedding Micheal Vito s youngest son and a decorated WW Marine is also present at the wedding Micheal seems to be uninterested in being a part of the family business Vito is a powerful man and is kind to all those who give him respect but is ruthless against those who do not But when a powerful and treacherous rival wants to sell drugs and needs the Don s influence for the same Vito refuses to do it What follows is a clash between Vito s fading old values and the new ways which may cause Micheal to do the thing he was most reluctant in doing and wage a mob war against all the other mafia families which could tear the Corleone family apart

Step2: Lower case

the godfather don vitocorleone is the head of the corleone mafia family in new york he is at the event of his daughter s wedding michealvito s youngest son and a decorated ww marine is also present at the wedding micheal seems to be uninterested in being a part of the family business vito is a powerful man and is kind to all those who give him respect but is ruthless against those who do not but when a powerful and treacherous rival wants to sell drugs and needs the don s influence for the same vito refuses to do it what follows is a clash between vito s fading old values and the new ways which may cause micheal to do the thing he was most reluctant in doing and wage a mob war against all the other mafia families which could tear the corleone family apart

Step3: Split

['the', 'godfather', 'don', 'vito', 'corleone', 'is', 'the', 'head', 'of', 'the', 'corleone', 'mafia', 'family', 'in', 'new', 'york', 'he', 'is', 'at', 'the', 'event', 'of', 'his', 'daughter', 's', 'wedding', 'micheal', 'vito', 's', 'youngest', 'son', 'and', 'a', 'decorated', 'ww', 'marine', 'is', 'also', 'present', 'at', 'the', 'wedding', 'micheal', 'seems', 'to', 'be', 'uninterested', 'in', 'being', 'a', 'part', 'of', 'the', 'family', 'business', 'vito', 'is', 'a', 'powerful', 'man', 'and', 'is', 'kind', 'to', 'all', 'those', 'who', 'give', 'him', 'respect', 'but', 'is', 'ruthless', 'against', 'those', 'who', 'do', 'not', 'but', 'when', 'a', 'powerful', 'and', 'treacherous', 'rival', 'wants', 'to', 'sell', 'drugs', 'and', 'needs', 'the', 'don', 's', 'influence', 'for', 'the', 'same', 'vito', 'refuses', 'to', 'do', 'it', 'what', 'follows', 'is', 'a', 'clash', 'between', 'vito', 's', 'fading', 'old', 'values', 'and', 'the', 'new', 'ways', 'which', 'may', 'cause', 'micheal', 'to', 'do', 'the', 'thing', 'he', 'was', 'most', 'reluctant', 'in', 'doing', 'and', 'wage', 'a', 'mob', 'war', 'against', 'all', 'the', 'other', 'mafia', 'families', 'which', 'could', 'tear', 'the', 'corleone', 'family', 'apart']

Step4: Removing stop words

['godfather', 'vito', 'corleone', 'head', 'corleone', 'mafia', 'family', 'new', 'york', 'event', 'daughter', 'wedding', 'micheal', 'vito', 'youngest', 'son', 'decorated', 'ww', 'marine', 'also', 'present', 'wedding', 'micheal', 'seems', 'uninterested', 'part', 'family', 'business', 'vito', 'powerful', 'man', 'kind', 'give', 'respect', 'ruthless', 'powerful', 'treacherous', 'rival', 'wants', 'sell', 'drugs', 'needs', 'influence', 'vito', 'refuses', 'follows', 'clash', 'vito', 'fading', 'old', 'values', 'new', 'ways', 'may', 'cause', 'micheal', 'thing', 'reluctant', 'wage', 'mob', 'war', 'mafia', 'families', 'could', 'tear', 'corleone', 'family', 'apart']

Step5: Stemming

['godfath', 'vito', 'corleon', 'head', 'corleon', 'mafia', 'famili', 'new', 'york', 'event', 'daughter', 'wed', 'micheal', 'vito', 'youngest', 'son', 'decor', 'ww', 'marin', 'also', 'present', 'wed', 'micheal', 'seem', 'uninterest', 'part', 'famili', 'busi', 'vito', 'power', 'man', 'kind', 'give', 'respect', 'ruthless', 'power', 'treacher', 'rival', 'want', 'sell', 'drug', 'need', 'influenc', 'vito', 'refus', 'follow', 'clash', 'vito', 'fade', 'old', 'valu', 'new', 'way', 'may', 'caus', 'micheal', 'thing', 'reluct', 'wage', 'mob', 'war', 'mafia', 'famili', 'could', 'tear', 'corleon', 'famili', 'apart']

Step6: Lemmatizing

['godfath', 'vito', 'corleon', 'head', 'corleon', 'mafia', 'famili', 'new', 'york', 'event', 'daughter', 'wed', 'micheal', 'vito', 'youngest', 'son', 'decor', 'ww', 'marin', 'also', 'present', 'wed', 'micheal', 'seem', 'uninterest', 'part', 'famili', 'busi', 'vito', 'power', 'man', 'kind', 'give', 'respect', 'ruthless', 'power', 'treacher', 'rival', 'want', 'sell', 'drug', 'need', 'influenc', 'vito', 'refus', 'follow', 'clash', 'vito', 'fade', 'old', 'valu', 'new', 'way', 'may', 'caus', 'micheal', 'thing', 'reluct', 'wage', 'mob', 'war', 'mafia', 'famili', 'could', 'tear', 'corleon', 'famili', 'apart']

7.5 Prediction

The most important step and the culmination of all efforts is the prediction step. For this task, several different classification methods are analysed and compared. The comparison is based on Accuracy, Precision, Recall and F1-score, calculated using 5-fold cross validation on the training set. In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k − 1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The models trained and tested are naive Bayes, random forest, SGDClassifier, Perceptron and Logistic regression. Several different parameters are evaluated for every model. For example, random forests with different number of trees (from 10 to 200) and trees of different depths (from 3 to 8) were tested. Same type of analysis follows for other classifiers and then comparisons on the aforementioned metrics are carried out. After a lot of experimentation and tuning, logistic regression was chosen as the classifier with the best results and also reasonable run time. To improve the results of logistic regression, classes are weighted for each genre prediction.

7.6 Results

The average accuracy ranges from 74% to 99% depending on the genre. The reason is that different genres have different counts in this data set and a model is trained for each genre separately, also some genres such as 'Adult' contain explicit words in their plot summaries that is not normally found in other genres' plot summaries; this increases the accuracy of our method to nearly 100% on this genre of movies. More details of results can be found in the table below (Only F-1 score is shown, since it's the harmonic mean of precision and recall, thus represents both):

Genre	Accuracy	Precision	Recall	F-1 score (average of 5 folds)
Short	86%	86%	86%	86%
Drama	78%	78%	78%	78%
Comedy	85%	84%	85%	85%
Documentary	94%	94%	94%	94%
Adult	100%	100%	100%	99%
Action	91%	92%	92%	92%
Thriller	90%	89%	90%	90%
Romance	90%	88%	89%	89%
Animation	96%	96%	97%	96%
Family	95%	95%	94%	95%
Horror	96%	96%	96%	96%
Music	97%	97%	97%	97%
History	96%	96%	96%	96%
Sport	98%	98%	98%	98%

It seems the hardest genres to predict are 'Drama' and 'Comedy'; probably because these genres embody a wide variety of plots. Besides, in the case of 'Comedy', ambiguous speech is common which could be throwing off the results.

```
genre = 'Action'
accs = list()
kf = KFold(len(train), n_folds=5)
print("             precision    recall  f1-score   support")
for train_folds, test_fold in kf:
    fit =  model.fit( train_data_features[train_folds], train[genre][train_folds])
    predict = fit.predict(train_data_features[test_fold])
    print(metrics.classification_report(train[genre][test_fold], predict,digits=3).split("\n")[5])
    acc = accuracy_score(train[genre][test_fold], predict)
    accs.append(acc)

print("total average accuracy: " + str(sum(accs)/len(accs)))
```

```
             precision    recall  f1-score   support
avg / total      0.915     0.917     0.916     24826
avg / total      0.916     0.919     0.917     24826
avg / total      0.919     0.921     0.920     24826
avg / total      0.921     0.923     0.922     24826
avg / total      0.915     0.917     0.916     24826
total average accuracy: 0.919302344316
```

Sample Statistics for 5-fold cross validation on the training set.

9. Clustering Using BoW

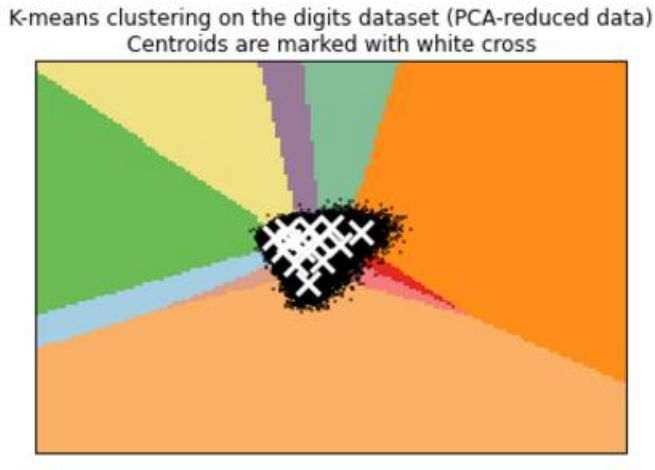
We also tried clustering the bag of words model using K-means clustering on the 14 genres present in themovieKeywordsPlots.list file. However, the results are not satisfying, the silhouette coefficient is ~0.01 which shows poor clustering.

Besides that, we also tried running PCA on this dataset and taking the first two principal components and plotting it (as found in an example on scikit's website). The PCA-reduced clusters are amassed in the middle; thus the method is also showing poor results.

Also, the clustering algorithm eats up a lot of memory, almost 12 GiB.

```
In [26]: from sklearn import metrics
         metrics.silhouette_score(train_data_features, estimator.labels_, metric='euclidean', sample_size=500)

Out[26]: 0.012319658563908266
```



The code can be found at the end of BoW3.ipynb.

10. Conclusion

In this project we have implemented the mining techniques that we learnt from the course. The IMDb dataset gave us a great opportunity to test out understanding of the mining techniques and test our application skills. We tried a lot of different methods for the mining process in order to improve the accuracy. In our opinion, with the dataset which we generated, it is not possible to get a better accuracy using the techniques that we learnt. We also tried to do something different and used NLP technique for genre prediction. The NLP technique did improve the accuracy of prediction. Overall, we are satisfied with our final results. We believe we have covered everything mentioned in the problem statement to the best of our abilities.

11. References

[1] Bo Li,1 Yibin Liao,1 and Zheng Qin2. 'Precomputed Clustering for Movie Recommendation System in Real Time'. Journal of Applied Mathematics. Volume 2014 (2014), Article ID 742341, 9 pages

For Clustering, Association rule mining and Classification:

- http://stats.stackexchange.com/questions/24540/categorical-variables-clustering-with-r
- http://www.inside-r.org/r-doc/cluster/daisy
- http://www.inside-r.org/packages/cran/klaR/docs/kmodes
- http://www.hindawi.com/journals/jam/2014/742341/- "**Movie Recommendation in Real Time"**
- https://shapeofdata.wordpress.com/2014/03/04/k-modes/

- http://www.inside-r.org/packages/cran/e1071/docs/naivebayes
- https://stat.ethz.ch/R-manual/R-devel/library/base/html/attributes.html

For Bag of Words method:

- https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-1-for-beginners-bag-of-words
- http://fastml.com/classifying-text-with-bag-of-words-a-tutorial/
- https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews
- https://www.kaggle.com/c/word2vec-nlp-tutorial
- http://scikit-learn.org/stable/modules/feature_extraction.html
- http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html