LEBANESE AMERICAN UNIVERSITY

Pre-Production Movie Rating Prediction Using Machine Learning

By

Firas Abdallah Gerges

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science

School of Arts and Sciences
November 2017

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

THESIS APPROVAL FORM

Student Name: Firas Gerges I.D. #: 20120287

Thesis Title : Pre-Production Movie Rating Prediction Using Machine Learning

Program: MS in Computer Science

Department: Computer Science and Mathematics

School: Arts and Sciences

The undersigned certify that they have examined the final electronic copy of this thesis and approved it in Partial Fulfillment of the requirements for the degree of:

MS in the major of Computer Science

Thesis Advisor's Name	Danielle Azar, Ph.D	Signature	[redacted]	DATE: 29 / 11 / 17 (Day / Month / Year)
Committee Member's Name	Haidar Harmanani, Ph.D	Signature		DATE: 29 / 11 / 17 (Day / Month / Year)
Committee Member's Name	Nashat Mansour, Ph.D	Signature		DATE: 29 / 11 / 2017 (Day / Month / Year)

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

THESIS COPYRIGHT RELEASE FORM

LEBANESE AMERICAN UNIVERSITY NON-EXCLUSIVE DISTRIBUTION LICENSE

By signing and submitting this license, you (the author(s) or copyright owner) grants the Lebanese American University (LAU) the non-exclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic formats and in any medium, including but not limited to audio or video. You agree that LAU may, without changing the content, translate the submission to any medium or format for the purpose of preservation. You also agree that LAU may keep more than one copy of this submission for purposes of security, backup and preservation. You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant LAU the rights required by this license, and that such third-party owned material is clearly identified and acknowledged within the text or content of the submission. IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN LAU, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT. LAU will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Name: Firas Geroges

Signature: [redacted]

Date: 29/11/2017

LAU
الجامعة اللبنانية الأميركية
Lebanese American University

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that:

1. I have read and understood LAU's Plagiarism Policy.
2. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
3. This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Firas Gerges

Signature: [redacted]

Date: 29/11/2017

To my loving parents

ACKNOWLEDGMENT

First and Furthermost, I want to express my deep gratitude to my thesis advisor, Dr. Danielle Azar. Without her endless support, motivation and hard work, the completion of this thesis would never be possible. I have acquired a lot of knowledge through her patience, guidance and instructions. Dr. Azar was always available whenever I bumped into a trouble or had a question about my research or writing. No words can describe the level to which I am grateful to this amazing doctor. Simply put, she is the friendliest, smartest, and the most hardworking supervisor one could ever wish to have.

Special thanks to my thesis jury members, Dr. Haidar Harmanani and Dr. Nashaat Mansour, for their time and valuable suggestions about this thesis.

I would like to offer my sincere gratitude to the Lebanese American University and the Department of Computer Science and Mathematics, for providing me with the required support, environment and equipment to complete my thesis. I would also like to thank the academic computer center administrator, Mr. Jalal Possik, for processing my requests.

I would also like to offer an exceptional appreciation to our academic assistant Ms. Samantha-Joe Beyrouthy for her endless support, motivation and help with all my departmental requests. Thank You Sam!

Working on a thesis, and doing graduate studies is not always delightful, and a person will often bump into difficulties. Without being surrounded by good friends, who inspirited, motivated and helped me overcome these difficulties, this work would never be possible. For this, I want to thank all the exceptional friends that I met during this amazing journey.

Last but the most important, I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Pre-Production Movie Rating Prediction Using Machine Learning

Firas Abdallah Gerges

ABSTRACT

Movie production is one of the most expensive investment fields and can result in enormous financial profit or loss. It is critical for investors and production companies to decide whether to invest in a certain movie given the huge loss that could occur from such investments. Hence, it is very beneficial to construct a model which helps investors in their decision making process. Machine learning has proven its effectiveness in building decision making models and recommender systems in various fields. In this work, we present several machine learning techniques (Support Vectors Machine, K-Nearest Neighbors, C5, Neural Networks and Case-Based Reasoning) along with a genetic algorithm to predict the success of a movie before its production using the IMDB rating as an indicator of the success. Results show that machine learning is useful in this domain and genetic algorithms can be used to build prediction models with relatively good performance.

Keywords: Machine Learning, Genetic Algorithms, IMDB, Classification, Data Mining, Forecasting, C5, Optimization, Predictive Model, Meta-Heuristics, Decision Making, Decision Tree, Instance-Based Learning, Neural Networks, SVM, Movies, Rating, Box-Office, Hollywood, Production, Casting.

`

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

The film industry, also known as the motion picture industry, has become a dominating industry in the entertainment field. According to the Internet Movies Database (IMDB) (http://www.imdb.com/), each year more than 20 thousand movies are produced, distributed and viewed worldwide. These numbers show the big impact of this industry on the market. Besides the cultural and sociological impact of movies, the motion picture occupies nowadays an important part of the business market with an average of 10 billion dollar worth of growth generated each year according to BoxOfficeMojo (http://www.boxofficemojo.com/).

The importance of the movie industry in our days does not rely only on the money generated, but also on the big number of people concerned with this industry. These factors make the production companies among the big economic players in their home country and important bodies in the international economy in general.

Given the importance of these companies, it is necessary to preserve their success and prevent their falling. In 2001, for example, Square Pictures released its first movie "*Final Fantasy: The Spirits Within*" (after its dedication to game development) which put the company in a very critical situation after investing 137 million dollars according to IMDB, and attaining a loss of 52 million dollars (Palmeri, 2013). The same situation was faced by Fox Studio, after releasing their animated movie "*Titan A.E*" which incurred a loss of 100 million dollars, the fact that forced FOX to close its animation studio and lay off hundreds of its employees

(Driver, 2016). Incidents like these occurred several times and across some major production studios and resulted in catastrophic failures. As such, it is very critical for a movie production company to decide whether or not to adopt the production of a certain movie. In order to make a suitable and safe choice, business, marketing, and economic strategies are used.

Simply put, adopting a movie means that the company is betting on this movie success. According to the viewer, a successful movie is a movie with high rating on the Internet, for example on IMDB. Those ratings are often the main factor which determines whether a person will eventually watch the movie. From this point of view, data analysts study the factors that can be used to determine the success of a movie. These are mostly pre-released factors. The success of movie is represented by either its rating or its box office growth. Such factors may include actors' names, writers, producers, genre, Production Company, Distribution Company and among others.

Combining all the above mentioned points, we conclude that it is necessary to develop a new technique to predict the success of a movie before its production. Such a technique will be a tool used by investors, to choose whether to adopt the production of a new movie.

In this work, we propose the use of machine learning techniques and implement a genetic algorithm to predict with relatively high accuracy movie user rating. In addition, we show that there is a correlation between a movie growth and its rating.

Our prediction strategy aims at finding the success status –represented by IMDB rating- of un-produced US movies using only information known before the production such as actors, producers, directors, writers, etc.

The rest of this thesis is organized as follows: In Chapter 2, we explore the previous studies related to movie's rating prediction. In Chapter 3, we discuss the data, its mining and processing. In Chapter 4, we introduce the various machine learning techniques to be used in this study. In Chapter 5, we explain the implemented approaches. Experimentation and results are presented in Chapter 6. Finally, we conclude in Chapter 7 and open the door for future work.

Chapter 2

Related Work

Previous work related to movie success can be divided into three main categories: 1. Work that studies movie features that affect its success. 2. Work that predicts user's preferences for movies, and 3. Work that predict the movie success. In the later, either the box office growth or the IMDB rating is used as success indicator.

(Hsu, Shen, & Xie, 2014) predicts users rating of movies using three different classification techniques: linear combination, multiple linear regression and neural networks. The authors do a comparison between the three used techniques based on the average error of each technique. Neural networks obtained a mean error of 69%, out-beating the other two techniques by 4% (linear combination) and by 12% (multiple linear regression). In (Latif & Afzal, 2016), the authors use some attributes (MPAA Rating, Awards, Screens, Opening weekend, Meta-score and budgets) to classify movies into *Terrible, Poor, Average,* and *Excellent* based on their IMDB user ratings. The authors use different classifiers: logistic regression, simple logistic, multilayer perceptron, J48, naïve bayes and PART and obtain classification accuracy of 84.15%, 84.5%, 79.07%, 82.42%, 79.66% and 79.52%, respectively. The authors in (Nithin, Pranav, Sarath Babu, & Lijiya, 2014) predict the gross revenue of movies as well as their IMDB rating using logistic regression, linear regression and support vector machine regression. The data is extracted from IMDB, Rotten Tomatoes and Wikipedia, and all the information is integrated in one database. Using backward greedy procedure, data is examined for correlation. Linear regression uses stochastic gradient descent to calculate the gross, which is the sum of all the attributes used,

multiplied each by some weight. Logistic regression is used by splitting the target variable into a number of groups of equal size and based on histogram representation of the movie revenue (drawing a histogram representing the revenue of movies in order to equally split this revenue into several groups). Results show that linear regression achieves the highest accuracy (50.7%) with 0.2 error tolerance. Logical regression reaches the lowest accuracy (39%) but a better error tolerance (0.125). SVM reaches the worst results for both the accuracy (39%) and the error tolerance (0.21). In (Saraee, White, & Eccleston, 2004), the aim of the work is to analyze movies parameters and their relation with the rating. Firstly, the authors perform a relevance analysis to detect factors that contribute most to highly rated movies. They detect the relation between film year and its rating by using a clustering technique. Finally, the authors classify the rating of upcoming movies using a universal classifier query. Similar to previous studies, the data used is from IMDB. After the relevance analysis, many attributes are eliminated from the studied set. The rating is classified into four categories: *excellent*, *average*, *poor* and *terrible*. Results show that the most important factors that affect the movie rating are the actors (90% relevance), directors (55%), budget (28%) and finally the genre (5%).

From another perspective, the work in (Oghina, Breuss, Tsagkias, & de Rijke, 2012) tries to predict a user movie rating using two type of features: surface and textual features from tweets and comments. Surface features include the number or volume of online activities around a movie (quantitative) and textual features refer to the meaning of those online activities (qualitative). In order to predict the rating, the authors implement a linear regression model. They run this model multiple times, using different features combination, and they compare the resulting error from each combination. The best result is from combining the tweeter features with the

likes/dislikes on Facebook. This combination gives a 42% mean absolute error and a 52% root squared mean error. The authors of (Tomar & Verma, 2015) predict and verify users rating of movies using a collaborative filtering system. The prediction strategy used consists of using previous rating attempts to identify the reliability of a user. This is done in order to ensure the correctness of movie rating. The next step is to evaluate unknown users to analyze their personality similarity to known users (collaborative filtering). This evaluation leads to group the users based on their movie type interest. Next, the authors run a similarity calculation to predict a user's rating using a group of similar users. Finally, they use a fuzzy inference process to improve the results of collaborative filtering when recommendation is unavailable. The authors try to filter users who rated more than the mean number of movies rated per day or they rated over twenty five percent of total users ratings. The authors calculate the similarity index and estimation. They use the collaborative filtering alone then combine it with the fuzzy system. Results show a noticeable improvement of the prediction square root error and the mean reliability when using the collaborative filtering and the fuzzy system combined, over using the collaborative filtering alone. (Saranya & Hussain, 2015) predicts a movie rating by a specific user using naïve Bayesian tree. Data is gathered from Movie Lens (https://movielens.org/). For each user, information includes name, age, gender, type of movie and questions that follow fuzzy rule. For the purpose of prediction, the authors calculate two indices: the cosine similarity which is the level of similarity between two users, and the predicted rating which is the prediction of the rating of a movie by a specific user. The Naïve Bayesian approach separates movies based on user's information, then separates the users based on their interests. The authors build a recommendation system that recommends to a certain user, a set of movies based on their predicted rating.

The authors of (Kawashima, Ogawa, & Haseyama, 2013) present a new method for rating prediction in e-commerce using an ordinal regression model based on a linear discriminant analysis. Data is collected from the reviews texts and images on the web; it includes keywords that appear in a defined number of reviews for each user. The authors give these key words weights using the tf-idf (term frequency–inverse document frequency) method (Sebastiani, 2002). The technique consists of component principle analysis, and then the extraction of HOG (histogram of oriented gradients) features for images. These represent appearance and shape of target objects. Next, the authors merge the textual and visual data using coefficient matrices. Finally, they use ordinal regression in order to predict the real rating. They tested this technique on movies' online reviews. An accuracy of 42% is achieved when using the multi-model features, one of 29% when using visual attributes and 30% when only textual data is used. In (Parimi & Caragea, 2013), the aim is to predict the pre-release box office success of movies after classifying them based on revenues. The data consists of movies described by actors, directors, runtime, genre release date, sequel information, and budget. Data is collected from BOXOfficeMojo. The authors classify the features of each movie into two groups: content features and link features. Link features capture dependencies between features and are used to form directed weighted graph. In the graph, two movies are connected if they have common actors or director or same genre or if they were released around the same time. The weighting scheme uses radical basic function capturing negative effects on edges. The classification in the directed graph is based on transductive algorithm[1] (Gammerman, Vovk, & Vapnik, 1998). Finally, the authors do the classification using matrix factorization. The highest accuracy

[1] Transductive algorithm consists of training an algorithm to test specific data in contrast to the inductive algorithm which uses training data to output general rules that can be applied on test data. For example, a case of a transductive algorithm could be when we want to train our algorithm to predict only the success of movies that have a "Drama" genre.

achieved was 33.56% using logistic regression. (Arundeep & AP, 2013) aims to predict the success class of the Indian movies using a mathematical model which uses neural networks. The work starts by collecting specific features of movies then assigning for each feature a specific weight which is the average value of this feature in the last ten movies. The authors use a neural network with different configuration of hidden layers to produce the best result in terms of true positive rate. Neural networks could achieve an accuracy of 93.3%. The work in (Borga & Hasbrouck, 2012) examines the impact of twenty nine variables on United States box office revenue, as well as the revenue to budget ratio to facilitate the decision making process of the producer or studio. The gathered data includes top grossing movies of 2010. The authors apply a correlation test to determine features that can be significant predictors of US box office. The technique used for this test is the ordinary least squares approach and the stepwise procedure. Results show that the budget, sequel, animation, sport, education and star power are significant predictors. In addition, the experiment indicates that R-rating[2] and Drama have a negative effect on the box office. Another significant attribute is the Horror genre which results in an average revenue to budget ratio of 6.517. The authors of (Shraddha, Hitarthi, & Darshana, 2015) integrate both the classical and social features of a movie to obtain better prediction accuracy. The data set contains features for 20 different movies. The authors classify the movies in this work manually as a hit or flop. Using J48 on WEKA (http://www.cs.waikato.ac.nz/ml/weka/), they obtained decision trees with testing accuracy equals to 66%. The study in (Sharda & Delen, 2006) uses neural networks to predict the financial performance of a movie at the box office before its theatrical release. Data was purchased from showbiz data (https://www.showbizdata.com). The

[2] R-rating movie is a movie that was rated as *Restricted* by the Motion Picture Association of America (MPAA).

authors use different types of independent variables, and then each variable is converted into a binary representation. Using the result of neural networks, the authors extract 2 different metrics: the percent of correct classification rate (Bingo) and the one away correct classification rate (1-away) which assumes the classification to be correct if there is only a difference of 1 between the real and the predicted rating. Using 10-folds cross validation, neural networks reached an accuracy equal to 36.9% on Bingo and 75.2% 1-away.

Given the big influence of online activities on prediction in general, the authors of (Mestyán, Yasseri, & Kertész, 2013) build a predictive model for the financial success of movies based on collective activity data of online users, in order to show that the popularity of a movie can be predicted in advance, by measuring and analyzing the activity level of editors and viewers and the corresponding entry of the movie in Wikipedia. Data used includes US movies with their corresponding Wikipedia pages. The complete dataset includes financial data as well as Wikipedia activity records. The first step in the model is to estimate the popularity based on four different activity measures: number of users, number of views, number of edits made and finally the collaborative rigor of the editing train (which is the same as the number of edits counting all the edits made by one user as one edit). Using these metrics, the authors calculate the Pearson correlation coefficient and use the results to predict the class label using the multi-variant linear regression model. The obtained coefficient of determination[3] R^2 is equal to 77%. The authors of (Bhave, Kulkarni, Biramane, & Kosamkar, 2015) suggest that the integration of both classical and social factors and the interrelation among the classical factors lead to a high accuracy. The data used includes the classical factors (actors, producers, genre, etc.) that are considered for the

[3] R-squared is the square of correlation between the predicted and the actual scores.

analysis of the movie. These are obtained from IMDB and similar websites. Moreover, the data includes social factors that contain: 1. sentiment analysis of the tweets made on Twitter, 2. YouTube view hits on the pre-movie videos (movie trailers), 3. the increment rate of the views as the movie release date approaches. A multivariate linear regression model is presented and this results in an R^2 of 70.57%. The work in (Barman, Chowdhury, & Singha, 2012) proposes a method that uses the back propagation neural network technique in order to predict a given movie profitability. The data used consists of twenty-two attributes. Among the used attributes, 20 represent genres after splitting them to several boolean attributes[4], and 2 attributes represent the popularity of the production company and the cast. The authors used binary attributes to specify whether a certain feature exists in the movie. The authors implement a back propagation neural network learning model using multi-layer feed forward. The model results in 87.5% accuracy on testing data. The authors in (Lash & Zhao, 2016) build a predictive model for movie success based on the cast collaborative network, the content and the time of the release. Data collected from BoxOfficeMojo includes different movies information. The authors split the attributes based on their type: the "what attributes" containing the rating, genre and the topic. The "who attributes" which include the star power and the collaborative network. The "when attributes" that include the average annual profit of the released data, the annual profitability in percentage by genre and an index called AWPG[5]. To classify movies in order to predict their profitability, the authors use a logistic linear regression, which obtains an accuracy of 77.1%. The work in (Wang, Cai, & Huang, 2010) investigates factors which influence innovation and imitation coefficient in new product diffusion

[4] For each genre, the authors specify whether it applies to the movie. For example *Horror*={0,1}.
[5] AWPG is an index defined by the authors and it includes the profit of similar movies along with the degree of similarity with the current movie.

model by introducing two forecasting models. Data collected describes movies released by major studios. Some of the related attributes are box office revenue, WOM (word of mouth), number of screens, critical reviews and others. According to this study, the probability of adoption of a new movie depends on innovation and imitation coefficients which are both calculated using specific equations. In order to classify new movies, the authors propose a mathematical model to estimate the two coefficients using influential pre-release information. Then, they use a stepwise regression with the two estimated coefficients to classify a movie into three different categories: *sleeper*, *mixed* and *blockbuster* strategies. The result of this model shows an accuracy of 60.6%.

(Zhang, Luo, & Yang, 2009) uses multi-layer back propagation neural networks to build a prediction model which classifies movies into six different categories ranged from *blob* (failure) to *bomb* (success). The authors gather data about box office movies from Wanda Cinema Line Company (http://www.wanda-group.com/businesses/culture/cinemas/) in China. After normalizing the variables, the authors give a weight from 0 to 1 to each attribute. The neural network used includes two hidden layers. The result of this technique gives a bingo accuracy of 68.1% and a 97.1% 1-Away accuracy.

In (Asad, Ahmed, & Saiedur Rahman, 2012), the authors propose a classification scheme of pre-released movies popularity using C4.5 and PART on WEKA. Data about movies is collected from IMDB and financial data from BoxOfficeMojo. The authors grouped the rating of the movies in four categories ranging from *excellent* to *terrible*. The two classifiers showed approximately the same results (with less than 1% variation). In order to deduce the factors that affect the rating, the authors generate two datasets. The first data set includes movie's information such as year, cast, etc. and the

second contains financial information. Results show that movie's content information affects the rating more than financial information. The accuracy of predicting movie rating based on content information is 77% as opposed to 56% when obtained from financial information.

The majority of the abovementioned work predicts the movie success after its production. Investors are, however, concerned with identifying the future performance of a movie before financing it, and hence, having a classifier that forecasts a movie performance after its making cannot have an effect on the decision making process and does not protect from possible financial losses. In our work, we try to fill this gap by predicting, with a relatively high accuracy, the success of a movie before its production using machine learning techniques and an instantiated genetic algorithm.

Chapter 3

Data Description

3.1 Data Source

IMDB is a widely known online database for movies and TV series. It offers data describing movies through its public interface (http://www.imdb.com/interfaces). Data is grouped into 43 files describing each a feature of the movie such as actors, directors, producers, etc. These are listed in Table 1.

Table 1 IMDB data files. The table shows the name of the file, its size in megabytes and a short description.

Name	Size (MB)	Description
actors.list.gz	295	The list of movies for each actor
actresses.list.gz	166	The list of movies for each actress
aka-names.list.gz	8.1	The list of name an actor/actress is known as
aka-titles.list.gz	9.1	The list of name a movie title is known as
alternate-versions.list.gz	2.5	the past editing for each show
biographies.list.gz	193	The biography of each actor/actress
business.list.gz	10.4	The box office growth and budget for movies
certificates.list.gz	5.6	The age restriction for each movie
cinematographers.list.gz	18.9	The list of movies for each photography director
complete-cast.list.gz	1	For each movie if list of cast is complete
complete-crew.list.gz	0.5	For each movie if list of crew is complete
composers.list.gz	15.0	The list of movies for each composer
costume-designers.list.gz	5.0	The list of movies for each costume designer
countries.list.gz	17.6	The origin country of each movie
directors.list.gz	34.0	The list of movies for each director
distributors.list.gz	26.7	The list of distributors for each movie
editors.list.gz	24.2	The list of movies for each costume designer
filesizes	0.01	The size of each file in the IMDB interface
genres.list.gz	17.0	The genres of each movie.
german-aka-titles.list.gz	0.3	The German translation of the titles
goofs.list.gz	21.3	The goofs in each movie

italian-aka-titles.list.gz	0.4	The Italian translation of the titles
keywords.list.gz	46.4	The list of keywords of each movie
language.list.gz	17.7	The language of each movie
literature.list.gz	6.1	Some literatures about movies
locations.list.gz	14.5	The location where each movie is shot
miscellaneous-companies.list.gz	14.9	The list of miscellaneous companies for each movies
movie-links.list.gz	22.5	The IMDB link of each movie
movies.list.gz	36.7	The list of movies and their year
mpaa-ratings-reasons.list.gz	0.3	The reason of MPAA rating for each rated movie
plot.list.gz	129	The plot of each movie
producers.list.gz	85.9	The list of movies for each producer
production-companies.list.gz	30.6	The production company of each movie
production-designers.list.gz	6.0	The list of movies for each production designer
quotes.list.gz	71.9	The quotes in each movies
ratings.list.gz	11.9	The IMDB rating of each movie
release-dates.list.gz	54.8	The release data of each movie
running-times.list.gz	14.8	The running times (in minutes) of each movie
sound-mix.list.gz	6.8	The sound mix company affiliated for each movie
special-effects-companies.list.gz	1.1	The special effect company of each movie
technical.list.gz	13.7	The technical information about the items used in production
trivia.list.gz	52.3	The list of trivia for each movie
writers.list.gz	53.5	The list of movies for each writer

3.2 Data Reduction

As a first step, we removed files that we consider irrelevant. A file is irrelevant if it contains information which is not directly related to the movie such as titles meaning, movies IMDB links, etc. Table 2 shows the list of files that we eliminated.

Table 2 Eliminated files that are irrelevant to movies.

File Name
aka-names.list.gz
aka-titles.list.gz
alternate-versions.list.gz
biographies.list.gz
complete-cast.list.gz
complete-crew.list.gz
filesizes.gz
german-aka-titles.list.gz
italian-aka-titles.list.gz
miscellaneous-companies.list.gz
movie-links.list.gz

Since the main aim of this work is to predict a movie success before its production for the sake of giving the investors a decision making technique to advise whether to invest in such a movie, all the features that we chose as indicators of success are pre-production information. Hence, we also eliminated all the files shown in

Table 3. These files contain after production information such as the mistakes in a movie production found by viewers (goofs), the certificates and MPAA ratings[6] earned by a movie after its production, a movie release date and its trivia, etc.

[6] MPAA ratings are the restrictions put on a movie concerning who can view it (e.g. Children, above 12, general public, etc.). These ratings are set by the Motion Picture Association of America (MPAA).

Table 3 Eliminated files that describe after-production movie information

File Name
goofs.list.gz
certificates.list.gz
mpaa-ratings-reasons.list.gz
quotes.list.gz
release-dates.list.gz
technical.list.gz
trivia.list.gz

The database contains a "Plot" file, which stores a description of each movie story. We also eliminated this file since the only interesting information we can extract from it is the key-words. These can be found in the "Key-words" file.

Due to the huge number of movies, we limited our study to US movies so we also eliminated all non-US movies from each file.

3.3 Processing and Cleaning

Due to the inconsistent format and patterns across files and within the same file, we developed a Java application to clean and reformat each file. The first file we cleaned is the list of movies "movies.list". We deleted any movie with an unknown year in addition to all the TV-series, non-US movies and short movies (running time less than

60 minutes)[7]. We included the year as part of the movie title in order to differentiate between movies having the same title but different years of production such as "*Within the Shadows 1991*" and "*Within the Shadows 2015*".

We extracted the list of genres for each movie from the "*genres.list"*. The next step was to delete all shows, news, and musicals. We stored this extracted information in a new file. Table 4 shows a subset of the new genre data file. Note that if a movie has multiple genres, the genres are all listed and separated by commas.

Table 4 List of movies with their genres

Title	Genres
The Elegant Clockwork of the Universe (2013)	Drama, Mystery, Sci-Fi
The Elektra/Vampyr Variations (2009)	Comedy, Fantasy, Horror
The Elementary Sherlock Holmes (2009)	Biography, Documentary
The Elements Club: Lord of Flawless Strength (2014)	Romance
The Elements Club: Unity Match (2015)	Romance
The Elements of Me (2016)	Reality-TV
The Elements of Me; Introducing Chosen Wilkins (2016)	Comedy
The Elephant King (2006)	Drama, Romance

Table 5 shows part of the rating file "*ratings.list*" after re-formatting it and excluding all the unwanted information such as IMDB movie ID, etc.

[7] The countries file -"*countries.list*"- contains the country of origin of each movie. We used this file to exclude all non US-movies from the "*movies.list*". In each used file, we deleted all the series.

Table 5 List of movies with their IMDB rating (rating over 10)

Title	Rating (/10)
American Pie (1999)	7.0
American Pie 2 (2001)	6.4
American Pie Presents Band Camp (2005)	5.0
American Pie Presents Beta House (2007)	5.3
American Pie Presents The Naked Mile (2006)	5.1
American Pie Revealed (2003)	6.8

We combined the resulting set of files that includes *rating, year of production, genre* and *running time* into one file. Table 6 shows an excerpt of this file.

Table 6 List of movies with their year of production, genres, rating and running time

Name	Year	Genre	Rating (/10)	Runtime (mins)
Painted Faces (1929)	1929	Crime, Mystery	5.5	74
I've Got Your Number (1934)	1934	Comedy, Romance	6.6	69
Bambi (1942)	1942	Animation, Drama, Family	7.4	70
The Strongest Man in the World (1975)	1975	Comedy, Family, Fantasy	5.9	92
Blue Ecstasy in New York (1980)	1980	Adult, Drama	6.0	90
Cheetah (1989)	1989	Adventure, Family	6.1	83
Adventures of Buttwoman (1991)	1991	Adult	6.8	90
No Escape No Return (1993)	1993	Action	3.8	91
Playback (1997)	1997	Adult	4.3	120
Love American Style (1999)	1999	Comedy, Romance	6.9	60
The Tooth Fairy (2006)	2006	Horror, Thriller	4.6	89

Next, we extracted the remaining features (*actors*, *production companies*, *directors*, *writers*, etc.) from their associated files and added them to the final data set. The link between the different files was the *movie title*. Given the actors/actresses files, it was impossible for us to extract from it, the primary actors/actresses for each movie. The files list all the available actors/actresses (from IMDB database) in alphabetical order, each followed by the list of movies he/she played a role in. For this, we used a free public API called "OMDBAPI" (http://www.omdbapi.com/). This API allows one to send a data request containing a movie name and its year of production. The Json response contains the primary actors/actresses. An example of the process is shown in Figure 1.

Request to get information about "Dawn of Justice 2016":

```
http://www.omdbapi.com/?t=dawn+of+justice&y=2016
```

The following is part of the Json response:


```
{"Title":"Batman v Superman: Dawn of
Justice","Year":"2016","Rated":"PG-13","Released":"25 Mar
2016","Runtime":"151 min","Genre":"Action, Adventure, Sci-
Fi","Director":"Zack Snyder","Writer":"Chris Terrio, David S. Goyer,
Bob Kane (Batman created by), Bill Finger (Batman created by), Jerry
Siegel (Superman created by), Joe Shuster (Superman created
by)","Actors":"Ben Affleck, Henry Cavill, Amy Adams, Jesse
Eisenberg","Plot":"Fearing that the actions of Superman are left
unchecked, Batman takes on the Man of Steel, while the world
wrestles with what kind of a hero it really
needs.","Language":"English","Country":"USA","Awards":"5
nominations….
```


Figure 1 An example of a request to OMDBAPI

From this response, we only used the *actors* field (e.gs: Ben Affleck, Henry Cavill, Amy Adams, Jesse Eisenberg) in order to get the primary actors of each movie. The rest of the data was already extracted by us from the IMDB files (such as *directors*, *writers*, etc.). We developed a PHP script in order to send the requests to "OMDAPI" and extract actors/actresses for each movie and save the results.

With all the above mentioned data reduction, processing and extraction, we built our final data set which contains the following features: *Year, rating, number of votes (*from the rating file*), running time, genre, distributors, production companies, special effect companies, sound mix key words, actors, actresses, cinematographers, composers, costume designers, directors, editors, producers, production-designers and writers*. All missing values were replaced with *NA*. Note that if, for a particular movie, a feature had multiple instances, we separated these instances with commas (for example, when having multiple production companies for the same movie.).

Given that we had some textual type features, we replaced each value of these features, by its average rating. For example, for an actor *X*, we extracted all the movies where *X* had a role in, then we computed the average rating of these movies. We replaced *X* with this average rating in the dataset. This approach might give an unfair bias towards factors (actors, directors, etc.) that were in a single movie with few user rating votes and high rating. In order to avoid this bias, we eliminated from our dataset all movies having a number of votes less than one thousand similar to what is done in (Saraee, White, & Eccleston, 2004).

After performing the textual to numerical transformation and because attributes can have multiple values at the same time (multiple actors for the same movie, directors, writers, etc.), we replaced all these values with their average.

As a next step, we rounded to the nearest integer, the rating in order to have a classification problem rather than predicting continuous values.

Chapter 4

Background Study

4.1 Decision Tree Learning

Decision tree learning is a machine learning technique based on inductive inference algorithms (Quinlan J. R., 1986).

Several algorithms use decision tree learning for classification purposes. In this section, we start by describing what decision trees are, then we describe ID3. The decision tree learning algorithm that we use in our work.

4.1.1 Decision Tree

Given a set of instances, a decision tree classifies each instance following specific criteria. This method is used to predict, or approximate, target functions which are discrete-valued, and in this scope, each decision tree represents a learned function. A decision tree consists of a set of nodes each corresponding to a test (condition). If the test at node N_i passes, the test in a child node is performed. This continues all the way down to a leaf. Leaves in decision trees represent the class label. An example of a decision tree is shown in Figure 2, where the problem is to decide whether to watch a movie based on attributes: *genre, rating* and *running time*. From a decision tree, we can deduce hypotheses. In this example, one hypothesis is: "I can watch a movie if its *genre* is "action", and has a "normal" *running time*".

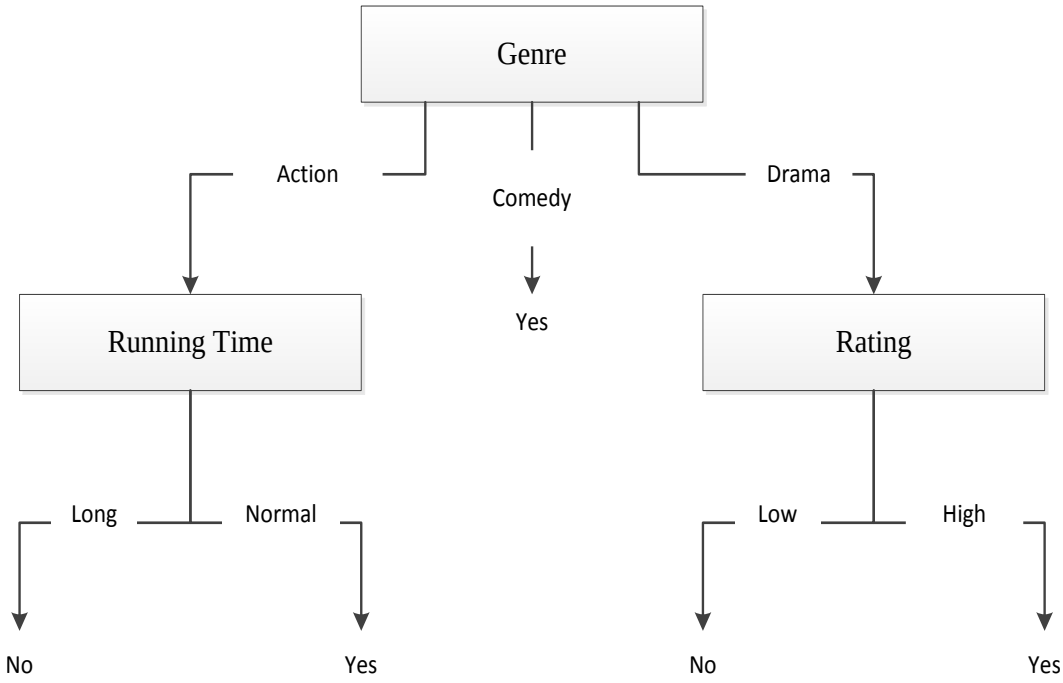

Figure 2 Decision tree to decide whether to watch a movie based on Genre, Runtime and Rating.

In the above mentioned decision tree, we have discrete-valued attributes (e.g. Rating: low, high), but in most cases, the attributes have continuous values. Given the very high number of different numerical values an attribute might have, it is impossible to put all these values individually in a decision tree. In Section 4.1.2, we see how to deal with such cases.

Decision trees can become very large and complicated which makes them hard to interpret by human experts. For this, in some domains, there is a need to transform them into rule sets. A rule set is formed of rules. A rule is a conjunction of conditions on attributes and a class label. For example, Figure 3 shows three rules extracted from the decision tree in Figure 2. A rule is formed by combining in one conjunction all conditions in a path from the root to a leaf node.

```
Rule 1:

If Genre = Action

Running Time = Long

Watch = No


Rule 2:

If Genre = Action

Running Time = Normal

Watch = Yes


Rule 3:

If Genre = Comedy

Watch = Yes
```

Figure 3 A rule set extracted from a decision tree

Rulesets can be pruned by eliminating some conditions. There are two pruning techniques: *Post-pruning* which consists of removing a condition only if this does not negatively affect the rule accuracy (potential to correctly classify data). Another type of pruning is called *Error-reduced Pruning* which prunes the tree directly before its transformation to a rule set. *Error-reduced Pruning* replaces a node in the tree by the most popular class label of its subtree. We keep this change only if the accuracy of the tree is not affected. For example, we replace the *Running Time* node in Figure 2 by "No" only if the accuracy of the tree does not deteriorate.

4.1.2 ID3

ID3 is one of the most widely used techniques in machine learning which uses decision trees to classify cases. The first step in ID3 is to choose the best classifier attribute.

The latter is defined by how much information is gained after splitting the tree on this attribute (Quinlan J. R., 1993).

For this, ID3 relies on a well-known concept in information theory called *entropy*. Entropy characterizes the purity or impurity of a set of data. In machine learning and prediction domain, the entropy refers to how well the data is unpredictable based on its target function. Our interest is to reduce the entropy of the data.

Given a data set, *S*, and a binary target function (+, -), the entropy of *S* is:

$$\text{Entropy}(S) = -P_{(+)} \lg P_{(+)} - P_{(-)} \lg P_{(-)} \qquad \text{Eq. 1}$$

where $P_{(+)}$ and $P_{(-)}$ are the proportion of the positive and negative values respectively. (lg is $\log_2$)

We can deduce from Eq. 1 that the value of the entropy is 0 when all the cases have the same class label (in which case, P_+ or P_- is equal to1) which means that *S* is easily predicted. This equation can easily be generalized when more than 2 class labels are in the data set (Eq. 2):

$$\text{Entropy}(S) = \sum_{c \,\epsilon\, ClassLabels} -P_c \lg P_c \qquad \text{Eq. 2}$$

The aim of ID3 is to choose the attribute which results in subsets with the lowest possible entropy in order to ease the process of classification. i.e. the subsets must be, each, as homogeneous as possible. For this purpose, ID3 computes, for each attribute, the expected reduction of entropy if the data set is split based on this attribute. This expected reduction is called *Information Gain* of attribute *A*.

The Information gain of *A* relative to a set *S* is given by the function below:

$$\text{Info}(S, A) = -\sum_{a \,\epsilon\, A} \frac{|S_a|}{|S|} Entropy(S_a) \qquad \text{Eq. 3}$$

The gain obtained from splitting *S* based on *A* is then computed as follows:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \text{Info}(S, A) \qquad \text{Eq. 4}$$

The attribute with the highest gain is then selected to from a node in the decision tree. The node encodes a condition consisting of *A* and a value[8]. The process is repeated for each node in the tree until all the attributes are fully exhausted.

To illustrate, consider the dataset shown in Table 7, where each case represents a movie characterized by its genre (*Comedy*, *Drama*, *Action*), its running time (*Short*, *Long*, *Average*), its budget (*Low*, *High*) and whether it was a success or not (1 = success, 0 = failure). ID3 first step is to calculate the entropy of the entire dataset.

$\text{Entropy}(S) = -3/6 \log_2 3/6 - 3/6 \log_2 3/6 = 1.$

In this case, the data set is equally divided based on the class label. Entropy of 1 indicates that 1 bit is required to classify a case (into 0 or 1).

Table 7 A data set of movies described by three attributes: Genre, Running time, Budget and class label indicating Success

ID	Genre	Running Time	Budget	Success
1	Comedy	Short	Low	1
2	Drama	Short	Low	1
3	Action	Long	High	1
4	Action	Long	Low	0
5	Action	Average	Low	0
6	Drama	Average	High	0

The next step is to calculate the information gain for each attribute. Starting with the *Genre* attribute, the data set can be divided into 3 subsets shown in Table 8.

[8] The value chosen in the condition is either a possible value of A or a cut point (explained in what follows).

Entropy (*Comedy*) = -1/1 $\log_2$ 1/1 – 0/1 $\log_2$ 0/1 = 0

Entropy (*Drama*) = -1/2 $\log_2$ 1/2 – 1/2 $\log_2$ 1/2 = 1

Entropy (*Action*) = -1/3 $\log_2$ 1/3 – 2/3 $\log_2$ 2/3 = 0.918

Info(S, Genre) = $-\frac{|S_{comedy}|}{|S|}\ Entropy(S_{comedy}) - \frac{|S_{drama}|}{|S|}\ Entropy(S_{drama}) - \frac{|S_{action}|}{|S|}\ Entropy(S_{action})$

$=-\frac{1}{6}*0-\frac{2}{6}*1-\frac{3}{6}*0.918$

= -0.792

Gain(S, Genre) = 1+ (– 0.792) = 0.208

Table 8 Data subsets after splitting the main set based on genre

Subset 1				
1	Comedy	Short	Low	1
Subset 2				
3	Action	Long	High	1
4	Action	Long	Low	0
5	Action	Average	Low	0
Subset 3				
2	Drama	Short	Low	1
6	Drama	Average	High	0

As for *Running Time*, the subsets are shown in Table 9.

Entropy (*Short*) = -2/2 $\log_2$ 2/2 – 0/2 $\log_2$ 0/2 = 0

Entropy (*Average*) = -0/2 $\log_2$ 0/2 – 2/2 $\log_2$ 2/2 = 0

Entropy (*Long*) = -1/2 $\log_2$ 1/2 – 1/2 $\log_2$ 1/2 = 1

Table 9 Data subsets after splitting the main set based on Running Time

Subset 1				
1	Comedy	Short	Low	1
2	Drama	Short	Low	1
Subset 2				
3	Action	Long	High	1
4	Action	Long	Low	0
Subset 3				
5	Action	Average	Low	0
6	Drama	Average	High	0

Info(S, Running Time) = $-\frac{|S_{long}|}{|S|}\ Entropy(S_{long}) - \frac{|S_{short}|}{|S|}\ Entropy(S_{short}) - \frac{|S_{average}|}{|S|}\ Entropy(S_{average})$

$=-\frac{2}{6} * 1 - \frac{2}{6} * 0 - \frac{2}{6} * 0$

= -0.333

Gain(S, Running Time) = 1 + (– 0.333) = 0.667

As for the *Budget* attribute, the subsets are shown in Table 10.

Entropy (*Low*) = -2/4 $\log_2$ 2/4 – 2/4 $\log_2$ 2/4 = 1

Entropy (*High*) = -1/2 $\log_2$ 1/2 – 1/2 $\log_2$ 1/2 = 1

Table 10 Data subsets after splitting the main set based on Running Time

Subset 1				
1	Comedy	Short	Low	1
2	Drama	Short	Low	1
3	Action	Long	Low	0
4	Action	Average	Low	0
Subset 2				
5	Action	Long	High	1
6	Drama	Average	High	0

Info(S, Budget) = $-\frac{|S_{high}|}{|S|}\ Entropy(S_{high}) - \frac{|S_{low}|}{|S|}\ Entropy(S_{low})$

$=-\frac{4}{6}*1-\frac{2}{6}*1$

= -1

Gain (S, Budget) = 1 – 1 = 0

Given the above calculated results, the attribute with the highest gain is the *Running time*. This attribute is then chosen by C5, to be the root of the tree by C5. The conditions correspond to each possible value of this attribute.

Then, at each step (for each child node) the above process is repeated on the subsets until the tree is fully constructed. Choosing the *Running time* results in three subset nodes (*Short*, *Long* and *Average*) as shown in Table 9. The above process (computing gains and choosing split attribute) is now repeated for each subset. Consider Subset 2 (S_2) of Table 9 (for *Running Time*=" Long").

Entropy (S_2) = -1/2 lg 1/2 – 1/2 lg 1/2 = 1

If splitting S_2 based on the *Budget:*

Entropy (*High*) = -1/1 $\log_2$ 1/1 = 0

Entropy (*Low*) = -1/1 $\log_2$ 1/1 = 0

Info (S_2, Budget) = $-\frac{|S_{2_{high}}|}{|S2|}\ Entropy\left(S_{2_{high}}\right)-\frac{|S_{2_{low}}|}{|S2|}\ Entropy\left(S_{2_{low}}\right)$

$=-\frac{1}{2}*0-\frac{1}{2}*0=0$

Gain (S_2, Budget) = 1

If splitting S_2 based on the *Genre:*

Entropy (*Action*) = -1/2 lg 1/2 -1/2 lg 1/2 = 1

Info (S_2, Genre) = $-\frac{|S_{2_{action}}|}{|S2|}\ Entropy\left(S_{2_{action}}\right)$

$= -\frac{2}{2} * 1 = -1$

Gain (S_2, Genre) = 0

The constructed tree so far is shown in Figure 4.

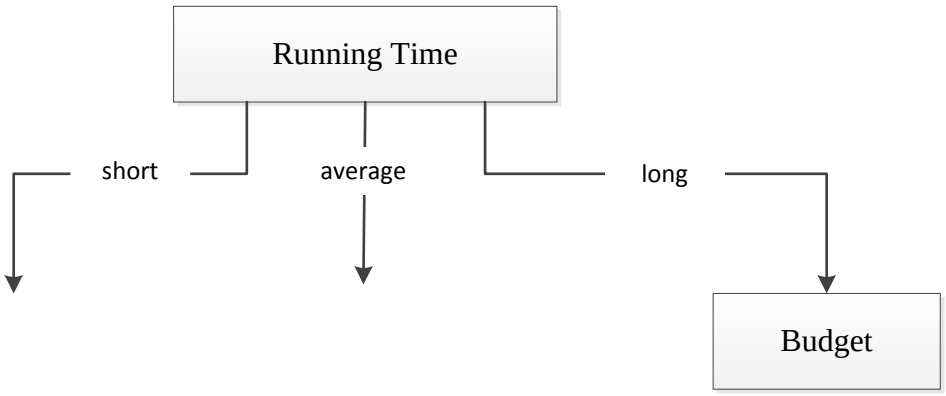

Figure 4 Partial decision tree constructed after two steps of C5 on the dataset presented in Table 2

C5, which is the latest descendent of ID3, provides the possibility to run the algorithm with a winnowing feature. The winnowing technique in C5 tries to reduce the number of used attributes before constructing the tree. This technique works by splitting the data randomly in two halves. The first half is used to construct a tree using the C5 normal process. The other half is used to prune the tree (Section 4.1.1). The pruned attributes are considered irrelevant and are referred to as the winnowed attributes. The final tree of C5 is constructed using the initial data set without the winnowed attributes.

For continuous attributes, the values are transformed to ranges by computing cut points (thresholds) for each attribute. These cut points are used to construct the test nodes. The pseudocode for computing thresholds is shown in Figure 5.

```
Cutpoints (Data):
Foreach attribute A in Data:
        Order(Data) based on A;
        Foreach case C_i in Data:
                if C_i.ClassLabel ≠ C_{i+1}.ClassLabel:
                        A.ThreshHold[i] = (C_i.A + C_{i+1}.A)/2;
```

Figure 5 Algorithm showing how to extract the thresholds form a data set "Data"

For example, suppose we had the dataset shown in Table 11, sorted on attribute *Running Time*. In this set, there are three different cut point values for attribute *Running time*. These are computed as the average of the two consecutive attribute values where the class label changes. Cut point 1 is (70+75)/2 = 72.5. Cut point 2 is (85+100)/2 = 92.5. Cut point 3 is (100+150)/2 = 125.

Table 11 Data set containing one attribute "Running Time", and a class label denoting whether to watch the movie

Running Time	**Watch**
50.2	1
70	1
75	0
85	0
100	1
150	0

An example of a decision tree constructed using continuous valued attributes is shown in Figure 6.

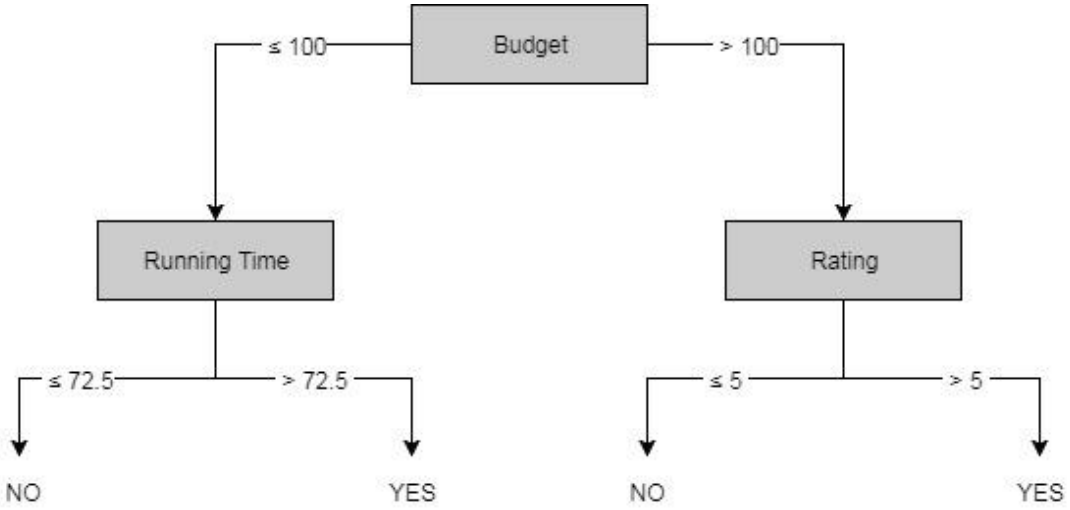

Figure 6 Decision tree for watching a movie or not based on Genre, Runtime and Rating as numerical data.

4.2 Artificial Neural Networks

Artificial Neural Networks (ANN) is a machine learning technique used to estimate, approximate or predict continuous, boolean and discrete valued functions (Mitchell T. M., 1997).

4.2.1 ANN Architecture

Inspired from human anatomy of neurons, neural networks are inspired from the interconnected neurons inside the human brain, forming different connected layers. ANN is formed of many electronically modeled neurons called artificial neurons.

ANN is based on a unit called a perceptron which calculates the linear combination of a vector of real-valued inputs and outputs 1 if this linear combination is greater than a threshold and -1 otherwise. Eq. 5 illustrate the functionality of a neuron. Where X_i and W_i represent the value and the weight of the ith attribute respectively. $-W_0$ represents the threshold.

$$O(X1,X2,\ldots,X_n) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} W_i X_i > 0 \\ -1 & \text{otherwise} \end{cases} \quad \text{Eq. 5}$$

Given a data set *A*, the goal is to find the best combination of weights whereby the perceptron outputs the correct label (-1 or + 1) for each instance in *A*.

. ANN usually contains three main parts: Input Layer, Hidden Layers and Output Layer, where each layer consists of non-connected units. The actual ANN can be viewed as a weighted directed graph as shown in Figure 7 where each unit in layer *K* is connected to each unit in layer *K+1*

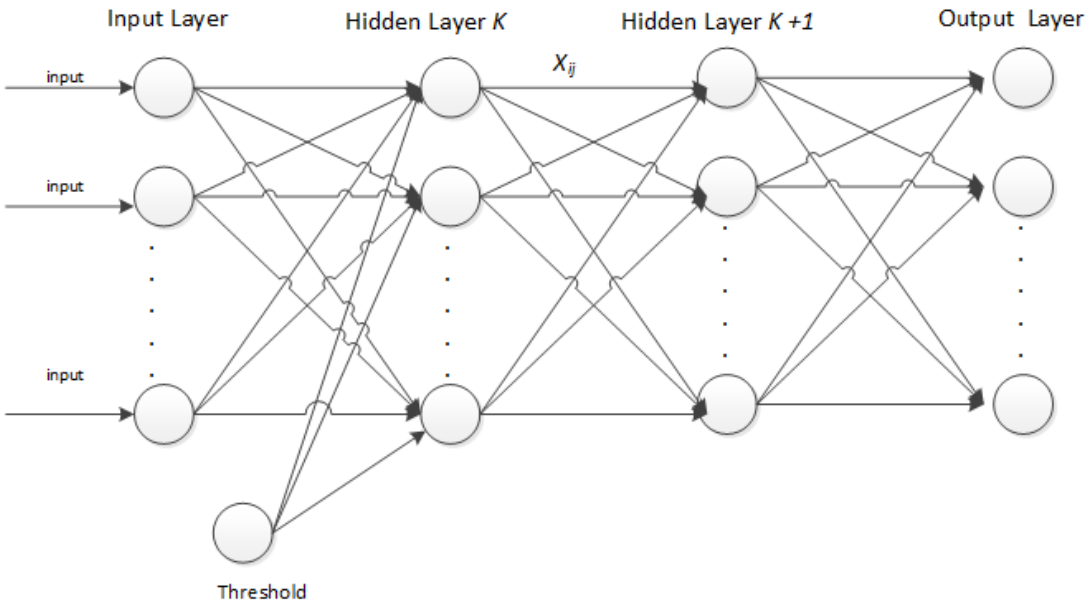

Figure 7 A Neural Network with two hidden layer. The Bias Input corresponds to the threshold.

The input layer in ANN consists of several units where each represents an attribute describing the data. One extra unit is added to represent the previously mentioned threshold. In ANN, each connection between two units (edge) is weighted. W_{ij} corresponds to the weight from unit *j* to unit *i*. The output of a unit *i*, denoted as *Xi*, will be an input for each unit in the next layer. The input of the first layer is the actual data. An input is denoted as X_{ij} which corresponds to the input of unit *j* from unit *i*.

The objective of the hidden layers is to compute functions whose output is forwarded as input for the next layer. In the output layer, the predicted or approximated output is calculated for each output unit using the perceptron. One output is then chosen as the final one based on certain criteria such as the majority value in the output layer.

4.2.2 ANN Algorithm

A Neural Network starts by initializing all weights to small random values, then computes the output of the perceptron and updates those weights based on that output. This process is repeated until all instances are classified correctly.

Updating the weights is based on what is called the *delta weight* which is computed after every classification attempts:

$$\Delta w_i = \eta (R-P) X_i \qquad \text{Eq. 6}$$

Where η is the learning rate and represents the degree to which the weights are changed, *R* is the real class label and *P* is the calculated class label by the perceptron. Hence, Δw_i is computed based on how far from the real class label, the predicted one is.

The above-mentioned process is suitable when the data is linearly separable. However, it does not guarantee convergence on nonlinearly separable data. In such a case, we can use the *delta rule* which uses the gradient descent[9] to obtain the weights that best fit the instances. The *delta rule* can be described as a perceptron without the addition of a threshold. In this scope, learning weights is done based on the training error which is:

[9] The gradient of a point is defined by the direction of its uphill.

$$E(w) = \frac{1}{2}\sum_i (R_i^{\,2} - P_i^2)$$ where *i* indicates an instance. Eq. 7

The gradient decent is used in order to find the weights that best decreases the error function *E (w)*. A Neural Network that uses gradient descent is shown in Figure 8.

ANN_Gradient_Decent (Training Instances):

Initialize all weights;

While status != termination_condition:

    Foreach unit i:

        Delta_W_i=0;

    Foreach training_instance <X, Y>:

        N=learning_rate;

        R=Y;

        Output=ComputeOutput(X) (Eq.5);

        $\Delta w_i = \Delta w_i + N(R\text{-}Outpout)$;

    Foreach unit i:

        $W_i = Wi + \Delta w_i$

Figure 8 Neural Network algorithm using gradient decent. (X, Y) is an instance in the training set. X=<x_1 …x_n> where x_i is an attribute and Y is the classification label.

4.2.3 ANN with Backpropagation

Another type of neural networks which can also learn non-linearly separable data uses the Backpropagation algorithm which has the ability to learn multi linear functions. In backpropagation, a *sigmoid* is used instead of the perceptron. The *sigmoid* is a nonlinear differentiable function that takes as input the *net output* ($\sum W_{ij}X_j$) and outputs: $\frac{1}{1+e^{-net}}$. In some implementation, the *tanh* function is used instead of the *sigmoid*.

The backpropagation algorithm is shown in Figure 9. For each training case, ANN with backpropagation computes in each unit the *net output* (Eq. 5) and apply the *sigmoid*

function on the results. After computing the output, this algorithm computes the error of each output and hidden unit based on the difference between the predicted and the real output. The final step is to update the weights accordingly.

ANN_Backpropagation (Training Instances):

Initialize all weights:

While status ≠ termination_condition:

Foreach training_instance <X,Y>:

Foreach unit i in NN:

Net_output=ComputeOutput(X) (Eq.5);

O_i = Sigmoid(Net_output);

Foreach output_unit j:

Error (j) = $O_j (1-O_j) * (Y - O_j)$;

Foreach hidden_unit i:

Error(i)= $O_i (1-O_i) * \sum_{j\ in\ output_{unit}} (Wi * ErrorOut(j))$;

Foreach W_{ij}:

Δw_{ij} = learning_rate* Error(i) * X_{ij};

$W_{ij} = W_{ij} + \Delta w_{ij}$;

Figure 9 Neural Network with Backpropagation

4.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a well-known machine learning technique that falls under the umbrella of instance-based learning (Mitchell T. M., 1997). The general idea of KNN is to classify a new case based on previously seen similar cases.

KNN defines similar cases of a new instance as the neighbors of this instance. In this technique, many approaches are being used to define similarity. The Euclidian distance

between cases is one such approach. The Euclidian Distance between two cases *A* and *B* is computed by the formula below:

$$E(A,B)=\sum_{i=0}^{n}\sqrt{(a_i - b_i)^2} \qquad \text{Eq. 8}$$

Where a_i and b_i represent the value of the ith attribute of *A* and *B* respectively, and *n* is the total number of attributes. Hence, two cases A and B are considered to be neighbors if they differ by little in their corresponding attributes.

In order to classify a new instance *N*, the algorithm works by computing the Euclidean distance as shown above, between *N* and each of the cases in the training set. The nearest *K* instances from the *Neighbors Set* are chosen and *N* is classified based on those instances. If the classification is discrete, *N* is assigned the majority class label shown in the *Neighbors Set*. If the classification is continuous, the average of the class labels in the *Neighbor Set* is computed and the result is assigned to *N*. Figure 10 shows the pseudocode of KNN.

KNN (Training Set, Testing Instances):

```
Foreach Testing_Case t:
      K_neighbors = t.Compute_Ecludian(Training_Set,K);
      If(Discrete):
      t.Classlabel = K_neighbors.get_most_existing_Classlabel;
      if(Continuous):
      t.ClassLabel = K_neighbors.getAverage();
```

Figure 10 KNN Algorithm

For example, suppose we have the movies shown in Table 12, where *title* represents the title of the movie, *running time* represents its total air time in minutes, *genre rating* represent the average rating of its genre[10] and finally the *rating* of the movie which is extracted from IMDB.

[10] The genre rating is calculated by computing the average rating of all movies having this specific genre.

Table 12 Movie Dataset

ID	Title	Running Time	Genre's rating	Rating
M_1	Craig Shoemaker: Daditude (2012)	82	6.98	7.4
M_2	Taking the Turn (2006)	134	5.35	6.9
M_3	Dead Last (2001)	60	6.33	7.8
M_4	Double Wedding (2010)	88	6.30	5.7
M_5	One Square Mile (2014)	96	6.28	6.3
M_6	Who Else to Blame? (2011)	97	4.80	3.7
M_7	Angel of Nanjing (2015)	70	7.21	7.9

Suppose furthermore we have the new movie with unknown rating shown in Table 13

Table 13 One Movie to classify by KNN

ID	Title	Running Time	Genre's rating	Rating
M_8	Batman v Superman: Dawn of Justice (2016)	151	5.75	?

In order to predict the rating of M_8 using KNN, the first task consists of computing the Euclidian distance between M_8 and M_{id} (id=1…7) shown in Table 12:

$$E(M_8,M_{id})=\sqrt{(RunningTime(M_8)-RunningTime(M_{id}))^2}+\sqrt{(Genre(M_8)-Genre(M_{id}))^2}$$

$$E(M_8,M_1)=\sqrt{(151-82)^2}+\sqrt{(5.75-6.98)^2}=70.23$$

$$E(M_8,M_2)=\sqrt{(151-134)^2}+\sqrt{(5.75-5.35)^2}=17.4$$

$$E(M_8,M_3)=\sqrt{(151-60)^2}+\sqrt{(5.75-6.33)^2}=91.58$$

$$E(M_8,M_4)=\sqrt{(151-88)^2}+\sqrt{(5.75-6.30)^2}=63.55$$

$$E(M_8,M_5)=\sqrt{(151-96)^2}+\sqrt{(5.75-6.28)^2}=55.53$$

$$E(M_8,M_6)=\sqrt{(151-97)^2}+\sqrt{(5.75-4.8)^2}=54.95$$

$E(M_8,M_7) = \sqrt{(151-70)^2} + \sqrt{(5.75-7.2)^2} = 82.45$

Given the above results, using 1NN, the nearest neighbor of M_8 is M_2, so the predicted rating of M_8 is 6.9.

Using different values for *K* will result in different classifications as shown below:

Using K = 2, the two nearest neighbors of M_8 are M_2 and M_6, and hence, the classification becomes:

2NN = $(R_2+R_6)/2$ = 5.3 where R_2 and R_6 are the ratings of M_2 and M_6 respectively.

Using K = 3, the two nearest neighbors of M_8 are M_2, M_5 and M_6, and the classification becomes:

3NN = $(R_2+R_6+R_5)/2$ = 5.6 where R_5 is the rating of M_5.

4.4 Support Vector Machine

Support Vectors Machine (SVM) is a supervised learning technique first introduced in Vladimir Vapnik's PhD thesis in the early 60s. The current version is invented by (Cortes & Vapnik, 1995). The cases are data points and the classification labels are binary. The aim of SVM is to find and build hyperplanes to split these data points based on their labels. Figure 11 shows two hyperplanes (H_1 and H_2) that correctly and precisely split the data points based on their colors where each color indicates a class label.

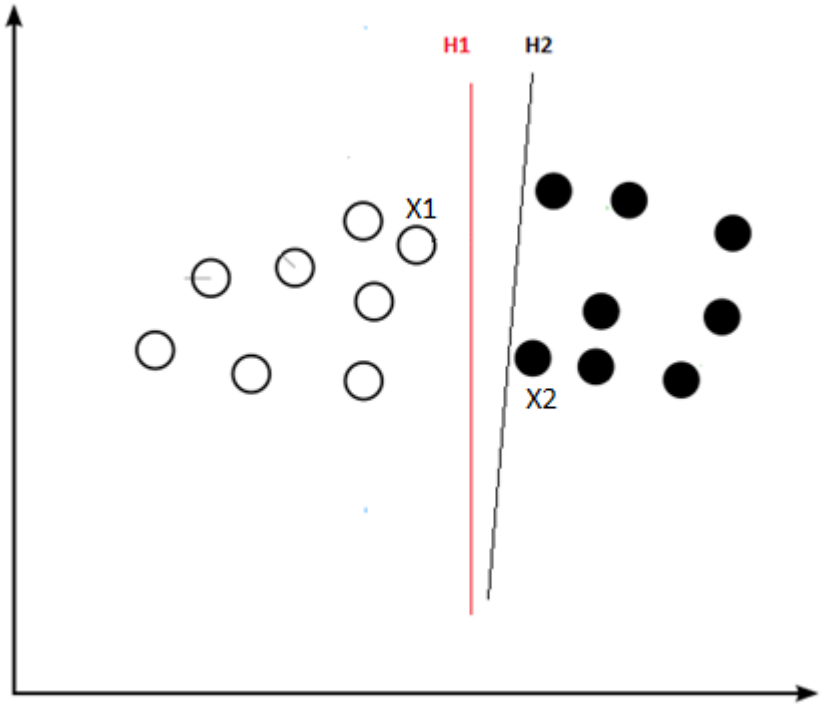

Figure 11 Data points separated by two hyperplane. Colors indicate class labels.

Since several hyperplanes may split these data points, the challenge is to know which hyperplane to choose. SVM looks for the hyperplane that has the biggest functional margin. The *functional margin* is defined as the distance between the hyperplane and the nearest data point (regardless of the class label). Hence, in Figure 11, SVM favors H_1 over H_2 since the distance between H_1 and the nearest data point X_1 is greater than the distance between H_2 and the nearest data point X_2.

Suppose we have *n* cases each labeled as positive (+) or negative (-). SVM looks for two hyperplanes H_+ and H_- which are the closest possible hyperplanes to the positive data points and the negative data points respectively (Figure 12). The area bounded by H_+ and H_- is called a "*street*". Any hyperplane (including the desired hyperplane H_{SVM}) that separates these data points will be located within the street region and it is desirable for H_{SVM} to have the largest distance to all data points, i.e. to be in the middle of the street. Hence, SVM aims at maximizing the distance between H_+ and H_- (the width of the street).

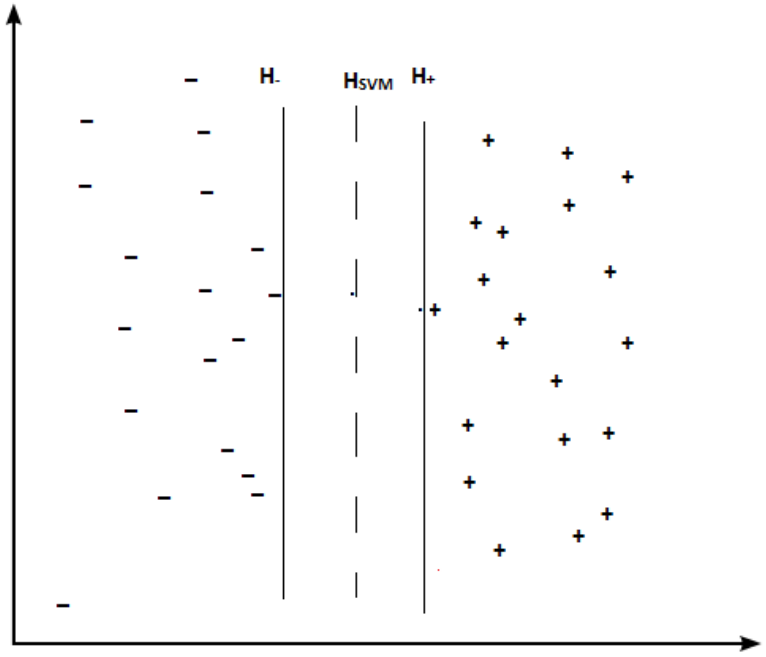

Figure 12 The H_+, H_- and the H_{SVM} as computed by SVM

Eq. 9 -11 show the equations of the hyperplanes H_{SVM}, H_+ and H_-, where x is vector representing a case, w a vector that is normal to the hyperplanes and b a constant.

$$H_{SVM}: w*x + b = 0 \qquad \text{Eq. 9}$$

$$H_+: w*x + b = +1 \qquad \text{Eq. 10}$$

$$H_-: w*x + b = -1 \qquad \text{Eq. 11}$$

Given an unknown case u, it is said to be positive if it is to the right of H_{SVM} and satisfies the following decision rule:

$$w*u + b \geq 0 \qquad \text{Eq. 12}$$

The problem consists of finding the optimal vector w such that, for all positive data points x_+, the following holds:

$$w*x_+ + b \geq +1 \qquad \text{Eq. 13}$$

And for all negative data points x_-:

$$w*x_- + b \leq -1 \qquad \text{Eq. 14}$$

In order to combine the above two constraints in one equation, a new variable y_i is used. It has the value +1 for positive cases and -1 for the negative ones. Multiplying the above two equations (Eq 13 and 14) by y_i, we obtain:

$$y_i * (w*x_i + b) \ -1 \geq 0 \qquad \text{Eq. 15}$$

For any x_i that belongs to the hyperplane, the following *Hyperplane Point Rule* applies:

$$y_i * (w*x_i + b) -1 = 0 \qquad \text{Eq. 16}$$

In order to maximize the width of the street we need an equation for this width. Suppose we have two points, x_1 and x_2, placed on H_+ and H_- respectively. The equations of these two data points are respectively:

$$w*x_1 + b = +1 \qquad \text{Eq. 17}$$

$$w*x_2 + b = -1 \qquad \text{Eq. 18}$$

Subtracting Eq. 17 and Eq. 18 and dividing by the magnitude of *w*, we obtain the following width equation:

$$\frac{w * (x_1 - x_2)}{|w|} = \frac{2}{|w|} \qquad \text{Eq. 19}$$

The aim is to maximize the width which is equal to $\frac{2}{|w|}$, thus, the problem is to minimize *w*, leading to:

$$\text{MAX } (2/|w|) \equiv \text{MIN } (|w|) \equiv \text{MIN } (½ \ |w|^2) \qquad \text{Eq. 20}$$

MIN (½ $|w|^2$) is used instead of MIN ($|w|$) because it is easier to minimize[11].

The problem now is to minimize an expression under a constraint (Hyperplane Points Rule (Eq. 16)). SVM tackles this problem using *Lagrange multipliers* as an optimization technique.

The *Lagrange multipliers* aims to maximize or minimize a given expression subject to different constraints by combining the initial expression and the given constraints in one single expression (Hand & Finch, 1998). In this case, it derives the main equation (Eq. 21) by subtracting all constraints (Hyperplane Points Rule (Eq. 16)) multiplied by a constant, α_i, from the objective function (½ $|w|^2$).

$$L = ½ |w|^2 - \sum \{\alpha_i * [y_i * (w * x_i + b) -1]\} \quad \text{Eq. 21}$$

Where α_i is called a *constraint multiplier*.

Finding the minimum value for *w*, consists of finding the value where the derivative of Eq. 21 is zero (Eq. 22).

$$\frac{dL}{dw} = w - \sum (\alpha_i * y_i * x_i) = 0 \rightarrow w = \sum (\alpha_i * y_i * x_i) \quad \text{Eq. 22}$$

$$\frac{dL}{db} = - \sum (\alpha_i * y_i) = 0 \rightarrow \sum (\alpha_i * y_i) = 0 \quad \text{Eq. 23}$$

Replacing *w* by its value from Eq. 22 in the initial decision rule (Eq. 12) of SVM, we obtain:

$$\text{If } \sum_{i=0}^{n} (\alpha_i * y_i * x_i) * x + b \geq 0 \text{ then } x \text{ is positive otherwise, it is negative.} \quad \text{Eq. 24}$$

Eq. 24 is the decision rule which is used to classify new unknown cases.

[11] Minimizing (½ $|w|^2$) consists of getting the values where its derivative is null, i.e. where w=0.

4.5 Case-Based Reasoning

Cased-Based Reasoning (CBR) is another instance-based learning technique, which, similar to KNN, depends heavily on the similarity between cases (Mitchell T. M., 1997). This technique is used widely in many different fields, like biomedical reasoning and law. Unlike KNN, it has to be instantiated depending on the problem at hand. The algorithm consists of four steps:

- Step 1: *Retrieve*. Consists of finding cases similar to the considered one. Similarity between cases is based on the attributes. For example, in the movie rating prediction problem, we say that movie *A* and *B* are similar if they have the same genre. So, a first step in the process of predicting a movie M_i ratingis to *retrieve* all movies form the training set (referred to as memory) having the same genre as M_i.
- Step 2: *Reuse.* Consists of using the retrieved similar cases in order to classify the new case M_i. For example, CBR may assign to M_i the rating of the similar case (if more than one similar case is *retrieved,* M_i is assigned the average rating of the retrieved cases).
- Step 3: *Revise*. Consists of revising the class label assigned to the new case, and changing it if different from what has been decided in Step 2.
- Step 4: *Retain.* Consists of adding M_i to the memory.

Table 14 shows a list of movies along with year of production, genre and their rating.

Table 14 Movies Dataset

ID	Title	Year	Genre	Rating
M_1	Journey from the Fall (2006)	2006	Romance	7.4
M_2	7-10 Split (2007)	2007	Comedy	4.6
M_3	Garfield: A Tail of Two Kitties (2006)	2006	Animation, Comedy, Family	5
M_4	Oh! My Dear Desire (2003)	2003	Action	6.9
M_5	Makin' Baby (2002)	2002	Comedy, Romance	4.3

To illustrate CBR, we consider M_i to be the case shown in Table 15.

Table 15 One case M_i, used for illustrating CBR

ID	Name	Year	Genre	Rating
M_6	Calendar Girls (2003)	2003	Comedy, Drama	?

The first step in CBR is to extract all similar cases. Movies M_2, M_3 and M_5 are similar to M_6, as they all share the comedy genre (Table 14). The next step is *Reuse*. We compute the average rating of all the similar movies:

Class_Label = (4.6 + 5 + 4.3)/3 = 4.6

The following step requires testing the new class label on the studied case. For example, consider that we assume that for each movie having "Drama" as one of its genres, this movie must have a rating greater than or equal to 6. Based on this hypothesis, we are forced not to accept the derived solution (4.6 < 6) in the *revise* step, and we can give the studied case a default label based on its attributes (genre= Drama) which is a rating of 6. The final step is to add this case along with the predicted rating (6) to the list of movies shown in Table 14.

4.6 Genetic Algorithms

Genetic Algorithms (GAs) are one of the most widely used meta-heuristics. Holland (Holland, 1975) invented this evolutionary algorithm in the 60's with the initial goal of studying the process of natural adaptation and using it in the domain of computer science (Mitchell M. , 1998). GA mimics the process of natural selection and survival of the fittest in order to find an optimal solution for a given problem (Goldberg & Holland, 1988).

GA starts by creating an initial population of chromosomes. Each chromosome represents a possible solution, and has a fitness value that describes the goodness of the corresponding solution. The fitter the chromosome, the higher its chance to survive. GA selects chromosomes based on their fitness from the current population, then performs certain genetic operations in order to create new chromosomes. The process continues until a desired solution is reached or another stopping criterion is reached. Normally, such criterion is problem-dependent. Figure 13 shows a flowchart describing the process of a GA.

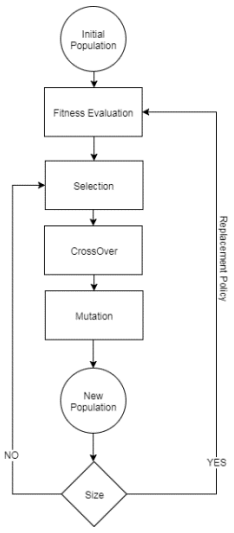

Figure 13 General Process of GA

4.6.1 Encoding Scheme

In biology, a chromosome is a sequence of genes, each represents a specific trait (e.g. the hair color). The most crucial part in GA is the encoding scheme which affects the performance of GA, and how fast it can find an optimal solution. The encoding scheme corresponds to the representation of chromosomes.

Initially, the representation of chromosomes relied on binary encoding where each gene had a value 0 or 1. Table 16 shows three chromosomes represented using the binary encoding.

Table 16 Three Chromosomes with binary encoding

Chromosome ID	Chromosome representation
C_1	1 0 0 0 1 0 0 1
C_2	1 1 1 1 1 1 0 0
C_3	1 0 0 0 0 0 0 1

Genetic Algorithms contain three core elements: population, fitness function and GA operators.

The population of chromosomes represents the search space of GA. This can be generated randomly or using another algorithm (e.g. greedy approaches or other heuristics).

The fitness function describes how well a solution is compared to others, and it is formulated based on the problem in hand.

The GA operators allow the GA to create new chromosomes from existing ones.

4.6.2 Genetic Operators

Typically, GA relies on two different genetic operations: "Crossover" and "Mutation". The former one consists of merging chromosomes by exchanging genes in order to create new ones. The latter one changes a given chromosome.

4.6.2.1 Crossover

Crossover is the core of the reproduction process, as it is the step that produces new chromosomes. In reproduction, crossover is the process in which two chromosomes are recombined to generate two new offspring by exchanging their genes. Each newly created offspring contains a combination of genes from both parents. Crossover occurs with a certain probability. The aim of crossover is to combine, in a single chromosome, the good traits that are spread across the population.

The most popular versions of crossover are: *1*-Point Crossover, *N*-Point Crossover and Uniform Crossover.

1-Point Crossover

1-Point Crossover is the first and most widely used version of this operator. It consists of cutting both parents at the same location. Then, the first offspring is formed by combining the first part of the first parent with the second part of the second parent. The second is created by combining the first part of the second parent with the second part of the first parent. Figure 14 shows the crossover between chromosomes C_1 and C_2 from Table 16.

Figure 14 Example of 1-Point Crossover. C1 and C2 are cut after the fourth gene.

N-Point Crossover

In this version of crossover, chromosomes are cut at more than one place (*N* places). Figure 15 shows a *2*-Point Crossover of C_1 and C_2. *N*-Point Crossover allows more changes in the created children as it consists of combining multiple parts from the parents.

Figure 15 Example of 2-Point Crossover, where C_1 and C_2 are cut after the third and the sixth genes.

Uniform Crossover

Under this form of crossover, a child inherits a particular gene from one of the parents randomly and the other child gets the corresponding gene from the other parent. Figure 16 shows an illustration of this type of cross over.

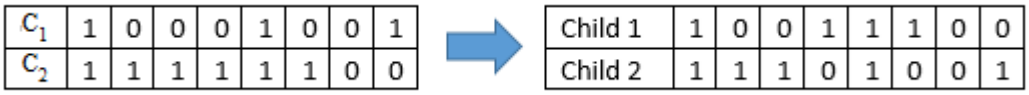

Figure 16 Example of Uniform Crossover where Child 1 inherits genes 1,2,3,5 and 7 from C_1 and genes 4, 6 and 8 from C_2. Child 2 inherits genes 4, 6 and 8 from C_1 and 1,2,3,5 and 7 from C_2.

4.6.2.2 Mutation

One limitation of crossover is that it does not create a complete new solution but combines previous ones. This fact may lead us eventually to local optimal solutions.

For example, given a population that contains only chromosomes that start with a “1”. Crossover alone creates only chromosomes that start with “1”. If the optimal solution start with a “0”, the GA will never find it. To overcome this problem, mutation can be used. Similar to crossover, there are several different mutation operators: mutation for binary representation, mutation for real-valued representation, mutation via swapping and inversion mutation.

In binary representation, the mutation consists of flipping the value of the gene from 0 to 1 or vice-versa.

Mutation for real-valued representation consists of replacing the gene value by a randomly chosen value from the set of the possible gene values (Figure 17).

Figure 17 Mutation for real-valued representation. Gene 7 is mutated

Mutation via swapping (Figure 18) consists of selecting randomly two genes and swapping their values. Inversion Mutation (Figure 19) consists of reversing a segment of genes.

Figure 18 Mutation via swapping. Gene 2 and Gene 5 are swapped

Figure 19 Inversion Mutation. The string in bold is reversed

4.6.3 Selection procedure

Since fitter chromosomes must have a higher chance of survival, the selection procedure is based on the fitness of a chromosome. Eq. 26 shows the selection probability for chromosome *i*.

$$S_i = \frac{f_i}{\sum_{j \epsilon P} f_j} \qquad \text{Eq. 26}$$

Where f_i is fitness value of chromosome *i* and *P* is the entire population of chromosomes.

The most widely used versions of this procedure are “Roulette Wheel Selection”, “K-Tournament Selection” and “Rank Selection”.

4.6.3.1 Roulette Wheel Selection

To illustrate this selection technique, we can imagine a roulette wheel divided among the chromosomes. Each chromosome is given a part of the wheel that is proportional to its fitness. A dice is thrown. The chromosome that receives the dice is then selected. Figure 20 shows this roulette wheel and its division among the chromosomes shown in Table 17. Obviously, the fitter the chromosome the higher its selection probability.

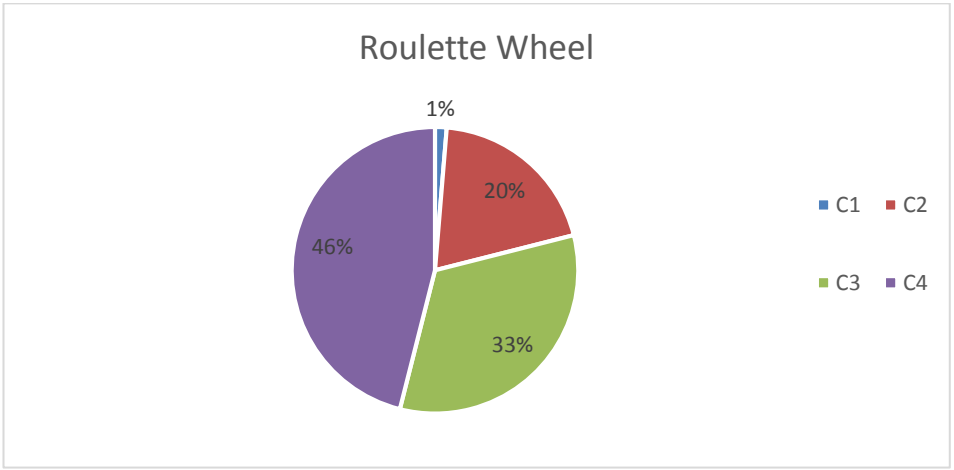

Figure 20 Example of a Roulette Wheel

Table 17 Chromosomes with their fitness values

Chromosome	Fitness Value
C_1	2
C_2	30
C_3	50
C_4	70

4.6.3.2 Rank Selection

Using roulette wheel selection, the chromosomes with very low fitness get a very little, if any, chance of being selected. As shown in Figure 20, C_1 only gets a 1% chance of being selected. To overcome this, rank selection ranks all chromosomes based on their fitness value (the fittest chromosome gets the highest rank), and then, roulette wheel is applied to the ranks rather than the finesses. Table 18 shows the chromosomes of Table 17 with their relative rank and Figure 21 shows their new roulette wheel allocation. Obviously, C_1 which had a minimal chance of being selected gets a better one now.

Table 18 Ranked Chromosomes

Chromosome	Fitness Value	Rank #
C_1	2	1
C_2	30	2
C_3	50	3
C_4	70	4

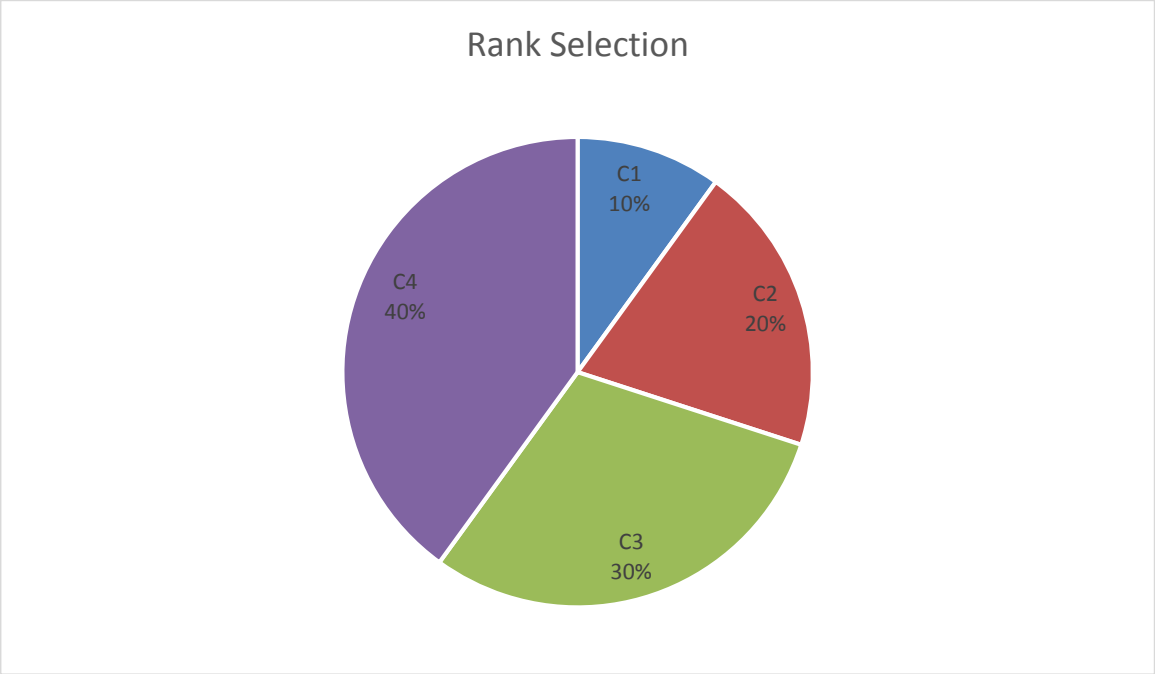

Figure 21 Roulette wheel allocated to chromosomes based on their ranks

4.6.3.3 *K*-Tournament Selection

K-Tournament Selection consists of selecting *K* chromosomes randomly from the population, choosing the fittest one as the first parent, returning the other *K-1* to the population, and repeating the process to choose the other parent (Figure 22).

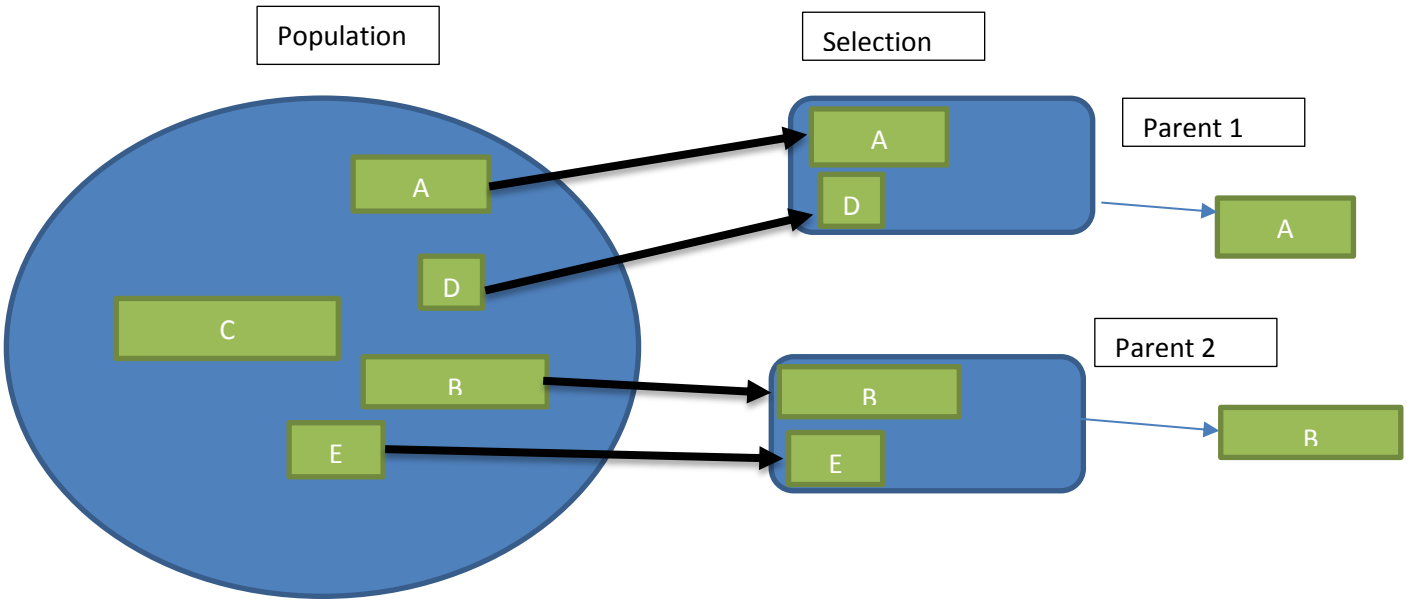

Figure 22 Illustration of 2-Tournament Selection

An advantage of *K*-Tournament selection is that chromosomes compete against each other in each subset and not in the entire population. This results in chromosomes with low fitness getting a higher chance of being selected. For example, in Figure 22, chromosome *A* is competing with *D* only which has a lower fitness. In other types of selections, *A* had to compete with the entire population including chromosomes *C* and *B* which both have higher fitness values. However, one disadvantage is that the chromosome with lowest fitness value in the population will never be selected. This is because this chromosome will always be compared with others having higher fitness.

4.6.4 Replacement Policy

The replacement policy defines which chromosomes from the current generation are e replaced. "*Generational GA*" and "*Steady-State GA*" are two different replacement policies.

Generational GA (Figure 23) is the first policy introduced by Holland. It consists of creating at each generation, a complete new population of the same size, composed of the newly generated offspring.

Steady-State GA (Figure 24) consists of keeping the same population, but replacing old, generally weak, chromosomes by the new offspring. This procedure is closer to what occurs in nature.

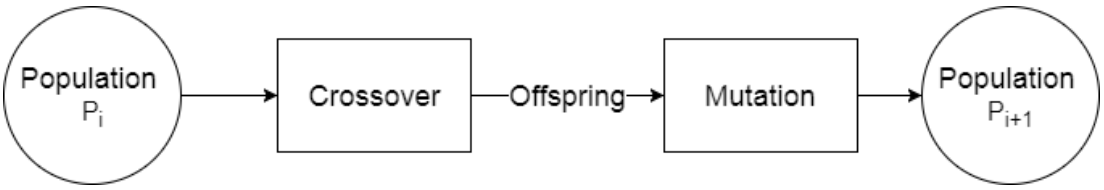

Figure 23 General Process of generational GA

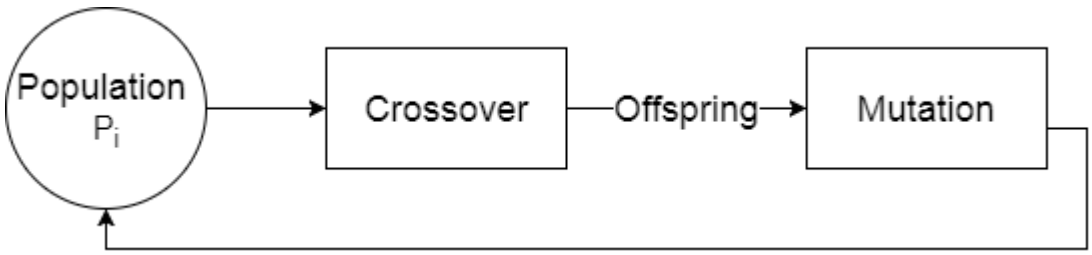

Figure 24 General process of a steady-state GA

One advantage of the Steady-State GA is that it allows new offspring to compete with their parents and other chromosomes as soon as they are created. Whereas, generational GA gives more diversity at each generation and hence is prone to converge to the optimal solution faster.

4.6.5 Elitism

Elitism is the procedure of copying the elite chromosomes from a generation to the next one. The aim of the procedure is to make sure that the best solutions found so far are not lost.

Chapter 5

Instantiation of Case-Based Reasoning and Genetic Algorithms

In this chapter, we explain how we instantiated the two techniques: Case-Based Reasoning and Genetic Algorithms.

5.1 Case-Based Reasoning

In our instantiation of CBR, we focus on the effectiveness of each attribute and the degree to which it affects the class label. As mentioned in Chapter 3, we transformed the textual data to numerical data by replacing each textual value by the average rating of all movies where this value appears. Hence, all attribute values –except *Running Time* and *Year* - are ranged between 1 and 10 (same as the rating). We define the degree of effectiveness of an attribute *A* in Eq. 25.

$$E(A) = \sum_{i \in Cases} \frac{1}{|v_i(A) - CL_i|} \quad \text{Eq. 25}$$

Where $v_i\ (A)$ is the value of attribute *A* in Case *i* and CL_i is the class label of Case *i*. This measures how close the value of *A* is to the class label.

As for the *Running Time* and *Year*, we gave them a 0 effectiveness degree because, in C5, they were eliminated after the winnowing technique. This indicates that they have very little effect, if any, on the classification.

Suppose we have the data set presented in Table 19. Applying Eq. 25, we can compute the degree of effectiveness of each attribute. Results are shown in Table 20 sorted in a descending order.

Table 19 Data set of movies used to illustrate CBR.

ID	Producer	Writer	Actor	Distributor	Rating
M_1	3.7	5.0	3.4	3.7	2
M_2	5.9	5.6	6.4	5.8	5
M_3	3.9	3.8	3.8	3.8	3
M_4	5.4	4.5	5.5	4.8	4
M_5	5.1	5.0	7.0	5.0	4
M_6	5.6	5.5	5.1	5.4	5
M_7	3.0	3.2	4.9	3.4	2
M_8	5.1	3.3	5.3	4.3	5
M_9	4.2	4.0	5.6	4.6	3
M_{10}	7.0	4.6	5.2	6.4	4
M_{11}	6.7	6.2	7.2	7.8	6
M_{12}	6.2	6.7	6.6	6.4	6
M_{13}	6.5	6.4	6.5	4.2	5
M_{14}	6.6	6.6	6.7	6.6	6
M_{15}	6.8	7.1	6.7	6.7	6
M_{16}	6.9	7.5	6.9	2.4	7

Table 20 Degree of Effectiveness of the attributes presented in Table 19

Attribute	Degree of Effectiveness
Actor	38.83765
Producer	33.6895
Writer	26.3224
Distributor	16.65783

The *retrieve* step consists of a series sub-steps, each called a *cycle,* aiming at retrieving the most similar cases based on the effectiveness of the attributes. It starts initially by sorting the attributes from the most effective to the least. Then, at each cycle, we consider the first attribute *A* in the sorted list, and, for each training case, we compute the difference between the value of *A* in this case and its value in the test case. We keep 5% of the cases with the least difference, then we go to the next cycle. The maximum number of cycles is equal to the number of attributes.

At the end of the *retrieve* step, we save the cases resulting from each cycle and move on to the *reuse* step.

The *rationale* behind the retrieve step, is that we are giving attributes with more effectiveness level, more weight when getting similar cases.

For example, we want to classify the movie M_{17} shown in Table 21.

Table 21 A test case

ID	Producer	Writer	Actor	Distributor	Rating
M_{17}	6.3	6.6	5.3	5.9	?

The *retrieve* step starts by sorting the attributes in descending order based on their effectiveness level (Table 20). In the first cycle (*Cycle 1*), we consider the first attribute in the sorted list: *Actor*. The next step is to compute the difference between the *Actor* attribute values of the training cases and the one of M_{17}. Table 22 shows an illustration of this process. Consider that we are keeping 50% of the training cases[12] that have the least difference. *Cycle 1* consists of the 8 cases (50% of 16 cases) having the least difference. These cases are highlighted in bold in Table 22.

As shown from Table 22, the most existing class label in *Cycle 1* is *5*.

The next cycle (*Cycle 2*) uses the next attribute on the list: *Producer*. Table 23 shows the difference between the *Producer* attribute values of the remaining training cases from *Cycle 1* and the one of M_{17}.

[12] In this illustration, we are keeping 50% of cases in each cycle given that the number of training cases used in the example is too small relatively the real number of training cases.

Table 22 Illustration of Cycle 1 of CBR. 50% of the cases having the least difference are highlighted in bold

ID	Actor	Actor of M_{17}	Difference	Rating
M_8	**5.3**	**5.3**	**0.0**	**5**
M_{10}	**5.2**	**5.3**	**0.1**	**4**
M_6	**5.1**	**5.3**	**0.2**	**5**
M_4	**5.5**	**5.3**	**0.2**	**4**
M_9	**5.6**	**5.3**	**0.3**	**3**
M_7	**4.9**	**5.3**	**0.4**	**2**
M_2	**6.4**	**5.3**	**1.1**	**5**
M_{13}	**6.5**	**5.3**	**1.2**	**5**
M_{12}	6.6	5.3	1.3	6
M_{14}	6.7	5.3	1.4	6
M_{15}	6.7	5.3	1.4	6
M_3	3.8	5.3	1.5	3
M_{16}	6.9	5.3	1.6	7
M_5	7.0	5.3	1.7	4
M_{11}	7.2	5.3	1.9	6
M_1	3.4	5.3	1.9	2

Table 23 Illustration of Cycle 2 of CBR. 50% of the cases having the least difference are highlighted in bold

ID	Producer	Producer of M_{17}	Difference	Rating
M_{13}	**6.5**	**6.3**	**0.2**	**5**
M_2	**5.9**	**6.3**	**0.4**	**5**
M_6	**5.6**	**6.3**	**0.7**	**5**
M_{10}	**7.0**	**6.3**	**0.7**	**4**
M_4	5.4	6.3	0.9	4
M_8	5.1	6.3	1.2	5
M_9	4.2	6.3	2.1	3
M_7	3.0	6.3	3.3	2

Cycle 2 consists of half of the remaining cases from *Cycle 1*. These cases are highlighted in bold in Table 23. The most existing class label in *Cycle 2* is *5*. Table 24 and Table 25 illustrate *Cycle 3* and *Cycle 4* respectively.

Table 24 Illustration of Cycle 3 of CBR. 50% of the cases having the least difference are highlighted in bold

ID	Writer	Writer of M_{17}	Difference	Rating
M_{13}	**6.4**	**6.6**	**0.2**	**5**
M_2	**5.6**	**6.6**	**1.1**	**5**
M_6	5.5	6.6	1.1	5
M_{10}	4.6	6.6	2.0	4

Table 25 Illustration of Cycle 4 of CBR. 50% of the cases having the least difference are highlighted in bold

ID	Distributor	Distributor of M_{17}	Difference	Rating
M_2	**5.8**	**5.9**	**0.1**	**5**
M_{13}	4.2	5.9	1.7	6

Table 26 shows the most existing class label in each cycle.

Table 26 Most existing class label of each cycle of CBR

Cycle	Most Existing Class Label
Cycle 1	5
Cycle 2	5
Cycle 3	5
Cycle 4	5

In the *reuse* step, we get the most frequent class label of each cycle from the *retrieve* step and save them in an array called "*Probable Class Labels*" (*PCL*) (Table 26). If the most existing class label in *PCL* is the same as the most existing class label of the cases in the last cycle, we give this class label to the new case, otherwise, we go to the *revise* step. In our case, the most existing class label in *PCL* is *5* which is the same as the most existing class label in the last cycle (*Cycle 4*). Hence, CBR assigns *5* to M_{17} as a class label.

The *revise* step is used whenever the most frequent class label in PCL is different from the most frequent class label of the last cycle. This step is performed to validate the classification. The *revise* step consists of removing the least effective attribute and

repeating the process starting from the retrieve step. This is done to ensure that attributes with low effectiveness, do not affect the prediction. The pseudo code of our CBR is shown in Figure 25.

```
CBR (Training_Cases, Testing_Cases):
Cases=Training_Cases
Effectiveness = Sort_Attributes_Based_On_Effectiveness (Eq. 25)
For each test_case in Testing_Cases:
        All_Cycles = Retrieve (Effectiveness, test_case, Cases)
        Class_ Label = Reuse (All_Cycles)
        If Class_Label == -1:
                Remove_Least_Effective_Attribute ()
                Repeat_All ()
Retrieve (Effectiveness, c, Cases):
//This function get similar cases of c from Cases based on the Effectiveness of attributes.
Cycle = 0
All_Cycles = [] // save the result of each cycle
While Cycle <= |Attributes| AND |Number_Cases| > 2:
        Attribute A = Effectiveness [0] // get first attribute
        Similar = get _5%_with_least_difference (A) //difference based on Attribute A
        Number_Cases = | Similar |
        All_Cycles [Cycle] = Similar
        Cycle = Cycle + 1
Return All_Cycles
Reuse (All_Cycles):
//This function computes the most existing class label of each cycle in All_Cycles and return the predicted class label.
PCL = []
For each cycle in All_Cycles:
        Cl = most_Class_Label (cycle)
        PCL.add (Cl)
If Last_Cycle_most_ClassLabel (Cycles) ==most_Class_Label (PCL):
        Return most_Class_Label (PCL)
Else:
        Return -1
```

Figure 25 The Pseudo-Code of the instantiated Case-Based Reasoning.

5.2 GA Instantiation

5.2.1 The Encoding Scheme

A chromosome represents a rule set. Each rule in the ruleset is a gene, i.e. a gene is a set of conditions along with a class label. The default class label is also a gene. Figure 26 shows a chromosome and the ruleset it represents.

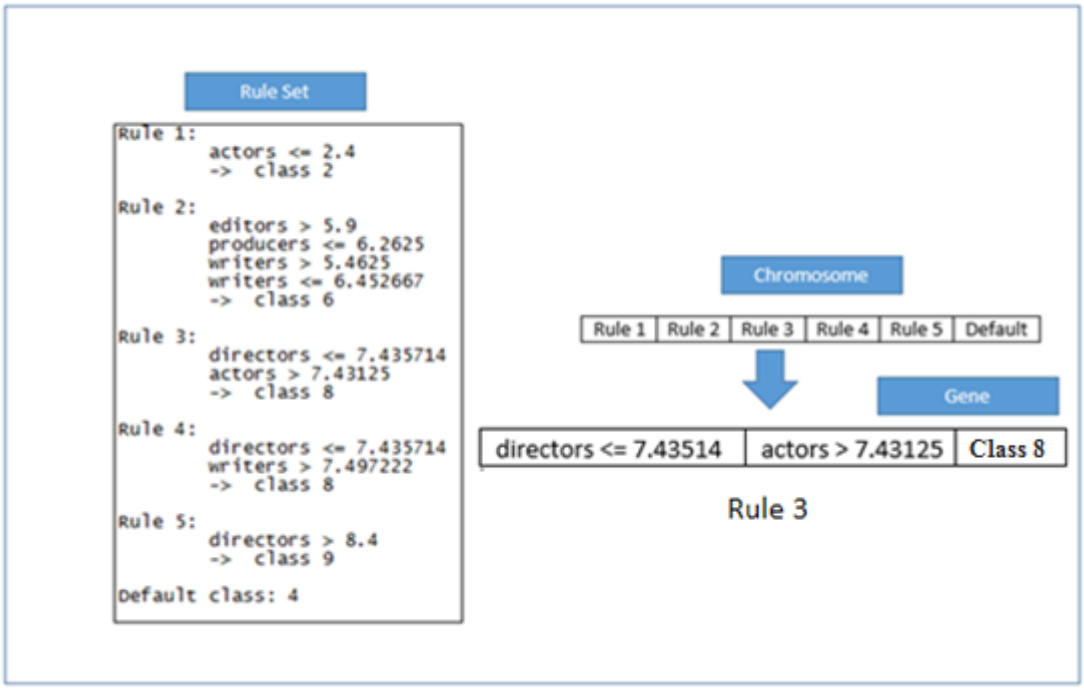

Figure 26 A ruleset and its corresponding chromosome

Our aim is to construct a new ruleset with a relatively good performance. Hence the prediction accuracy of a ruleset represents the fitness value of the corresponding chromosome.

The initial population consists of several rulesets created by C5 using the training data. In the next subsections, we describe the evolution process of the GA along with the different operators used.

5.2.2 Confidence Index

To describe and locate the good traits in chromosomes, we use the confidence index to describe how good a rule is. The confidence index, *CI,* of rule *R* is the total number of the cases which are correctly classified by *R* (*C(R)*) over the total number of cases covered by *R* (*Cov(R)*) [13].

$$\mathrm{CI(R)} = \frac{C(R)}{Cov(R)} \qquad \text{Eq. 27}$$

Suppose we have the data shown in Table 27, and the following rule *R*:

R: Actor ≤ 7, Class Label: 5

Table 27 Example of a dataset used to show how CI (R) is computed

Actor	Distributor	Writer	Class Label
5.2	8.7	5.7	5
4.3	6.9	6.5	5
6.2	9.4	3.5	7
8	2.5	8	8

Rule *R* covers the first three cases, and it correctly classifies only the first two. Hence, the confidence index of *R* is:

$$\mathrm{CI\ (R)} = \frac{2}{3}$$

5.2.3 Condition Dictionary

In order to explore new solutions, we create new conditions from the training set. These new conditions are saved in a "*Condition Dictionary*". To form these conditions, for each attribute *A,* we sort the cases in an ascending order based on the value of *A,* then we extract the values of *A* for which the class label changes from CL_{old} to CL_{new}.

[13] A case is covered by *R* if it satisfies all conditions of *R.*

For each extracted value *V*, we construct two conditions, each corresponding to one class label.

For example, given the data set shown in Table 28, the cases are sorted based on the value of *A* in an ascending order. The class label changes when *A* has a value of 6, hence "6" is the extracted value and the two constructed rules are:

1) A ≤ 4: class label 1
2) A > 4: class label 0

These conditions are saved in the *Conditions Dictionary*. We will see later how this dictionary is used in the mutation process.

Table 28 Dataset sorted based on the value of attribute A

A	B	C	Class Label
2	12	12	1
4	1	4	1
6	45	1	0

5.2.4 The Evolution Process

The pseudocode of the instantiated GA is shown in Figure 27. We start by applying elitism whereby a certain percentage (ELIT %) of the population is copied as is to the next generation. Then, two parents are selected and crossover is applied to create one child only. The child is mutated with probability μ and added to the new generation. The best of the two parents undergoes mutation as well with probability μ' and is copied to the next generation. This is repeated until size of generation G_{i+1} becomes equal to that of G_i. This process is repeated until we reach a certain plateau *MAXPLAT* or a maximum number of generations *MAXGEN*.

5.2.5 Selection

We use K-tournament as our selection technique (Chapter 4, Section 6).

```
Instantiated GA (MAXPLAT, MAXGEN, ELIT_PERCENT, μ, μ'):
P = C5.createRuleSets ()
Until MAXPLAT or MAXGEN:
        G=null
        G.add(bestChromosome(P), ELIT_PERCENT)
        While |G| < |P|:
                Parent1 = K_Selection (P)
                Parent2 = K_Selection(P-Parent1)
                Child = CrossOver (P1, P2)
                Child' = Mutate (Child, μ)
                BestParent=getBest (P1, P2)
                BestParent = Mutate(BestParent, μ')
                G.add(BestParent)
                G.add(Child')
        P=G
return bestChromosome(P)
```

Figure 27 Pseudo-Code of our instantiated GA

5.2.6 Crossover

We use a uniform crossover where we select a random gene (rule) from each parent and the rule with the higher confidence index between the two is copied into the child. This process might lead the GA into a local optimum given that it is biased towards choosing only the good traits form the parents. In order to solve this issue, the process of selecting the rule based on its confidence is done with a certain probability, *Pconfidence*, otherwise the rule is inherited with a fifty-fifty chance from parent 1 or parent 2. The pseudo code of crossover is shown in Figure 28.

```
CrossOver (P1, P2):
Child=new RuleSet
Child.defaultClassLabel = chooseRandomly (P1.defaultClassLabel,
P2.defaultClassLabel)
For i: 0...NumberofRules:
        If Pconfidence:
                Child.rules.add(chooseBest(P1.rules[i],P2.rules[i]))
        Else
                Child.rules.add(chooseRandomly(P1.rules[i],P2.rules[i]))
ComputeAccuracy(Child)
return Child
```

Figure 28 Pseudo-Code of crossover

5.2.7 Mutation

Mutation consists of adding, changing or removing conditions in a chromosome. To mutate a gene, we either add to it a new random condition from the *Conditions Dictionary* that has the same class label as the gene in hand, or a random condition from this gene is selected and it is either removed or its value is replaced by a random value from the *Conditions dictionary*. This mutation is important as it provides the GA with the ability to explore new conditions, hence new solutions.

5.2.8 Replacement Policy

We are using a generational GA, whereby the newly generated offspring are copied to the new generation.

Chapter 6

Experiments and Results

6.1 Data Sets

As already mentioned in Chapter 3, the cleaned data set consists of 18 attributes (Table 29), and 8 different class labels representing the movie IMDB rating (Ranging from 2 to 9)[14]. We performed our experiments on movies released between 2000 and 2017. The final data set consists of 4,883 movies.

Table 29 the attributes of the final dataset and their type

Attribute Name	Attribute Type
Actors	Continuous
Cinematographers	Continuous
Composers	Continuous
Costume-designers	Continuous
Directors	Continuous
Distributors	Continuous
Editors	Continuous
Genre	Continuous
Keywords	Continuous
Producers	Continuous
Production-Companies	Continuous
production-designers	Continuous
Runtime	Discrete
Sound-mix	Continuous
Special-effects-companies	Continuous
Writers	Continuous
Year	Discrete
Actors	Continuous

To deal with the missing values in the data, we replace in each case C_i , each missing value of attribute A_i by the average value of A_i of all the cases having the same class

[14] After rounding the ratings to the nearest integer, and removing all movies with less than 1000 user rating votes, the resulting data set did not include movies of rating 1 nor 10.

label as C_i. For example, suppose we have a movie *M* of rating 5 and with the *editor* attribute missing. We average all the *editor* values of all the movies having a rating of 5. This average is assigned to *M* as a value for its *editor*.

6.2 Experiments

We perform 10 fold cross-validation (Breiman, Friedman, Stones, & Olshen, 1984). We split the initial data set randomly into 10 different folds of approximately equal size. For each fold *Fi*, we train our algorithms on the combination of the other 9 folds *Fj* (for every $j \neq i$) and we test on *Fi*. We compute the training accuracy and the testing accuracy for each *i*. We report the average accuracy along with standard deviation on both training and testing data (Table 30). We also include the results of the random and the majority classifiers[15]. We highlight in bold the highest obtained results in term of average testing and training accuracy, and in italics the lowest results other than the random or major classifier. Figure 29 and Figure 30 show the same results in graphs.

Table 30 The performance of each machine learning techniques on training and testing data

Technique	Training Accuracy (std)	Testing Accuracy (std)
C5	88.9 (1)	85.3 (1.8)
Neural Networks	89.15 (0.9)	76.8 (2.8)
KNN	*84.1 (0.2)*	*69.6 (1.5)*
SVM	87.3 (0.2)	78.3(1)
CBR	**99.6 (0.02)**	85 (1)
GA	97.2 (0.6)	**90.5 (0.9)**
Majority Classifier	32.1 (0)	32.1 (0.04)
Random Classifier	12.6 (0.4)	12.8 (0.6)

[15] The majority classifier assigns to new cases the majority class label in the training data breaking ties randomly. The random classifier consists of assigning a random class label to new cases.

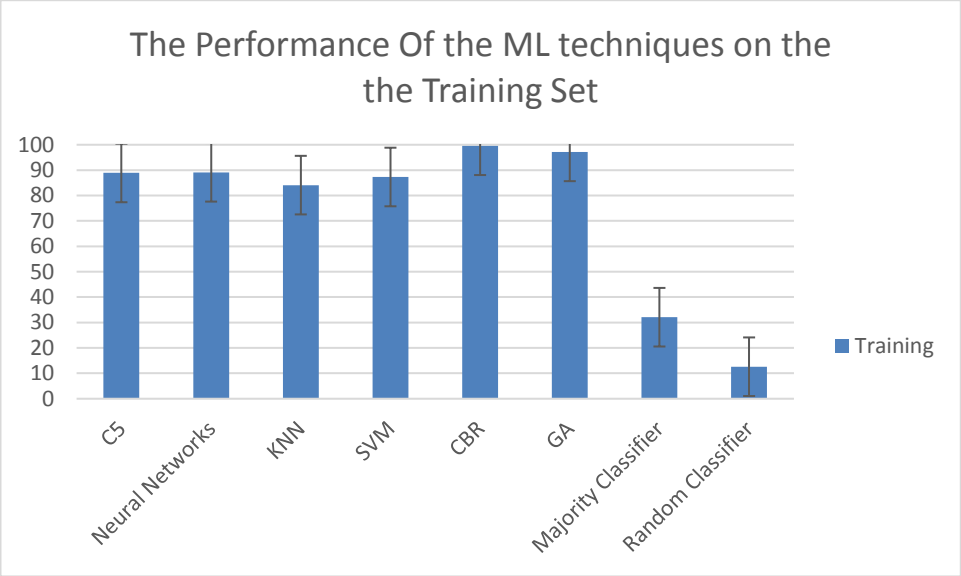

Figure 29 The performance of the ML techniques on the training set

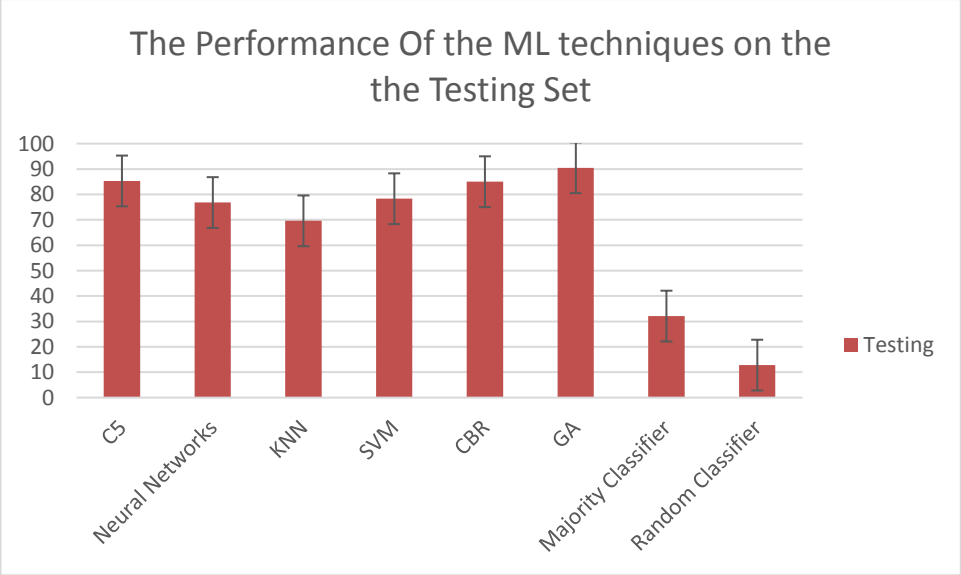

Figure 30 The performance of the ML techniques on the testing set

6.2.1 ID3

We ran C5 (the latest descendant of ID3, available as a source code from Rulequest (www.rulequest.com)) on the 10 folds with winnowing. Results in Table 30 show that there is no data overfitting and the technique is stable (over the different 10 folds) as

it resulted in a low standard deviation. We also noticed that C5 shows the best performance following our GA.

6.2.2 Artificial Neural Networks

We applied a multilayer Neural Networks with backpropagation which is free to use using Weka (http://www.cs.waikato.ac.nz/ml/weka/). Four parameters affect the classification performance of Neural Networks: the learning rate, number of hidden layers, number of nodes in a layer and maximum number of iterations. In order to obtain the most optimal combination of parameters, we ran Neural Networks multiple times, fixing 3 parameters each time and changing the fourth. The charts in Figure 31, Figure 32 and Figure 33 show the performance of the Neural Network and how it changes with the used parameters.

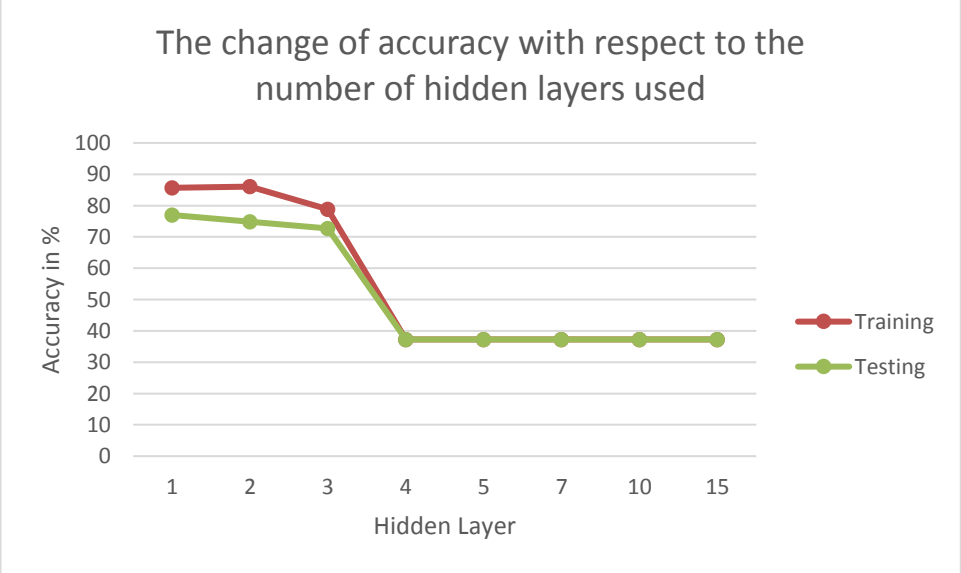

Figure 31 A chart describing how the training and testing accuracies of NN change with the number of hidden layers. Learning rate = 0.05, Number of Nodes =45, Number of iterations =750

Figure 31 shows that the effectiveness of the backpropagation decreases when we increase the number of hidden layers.

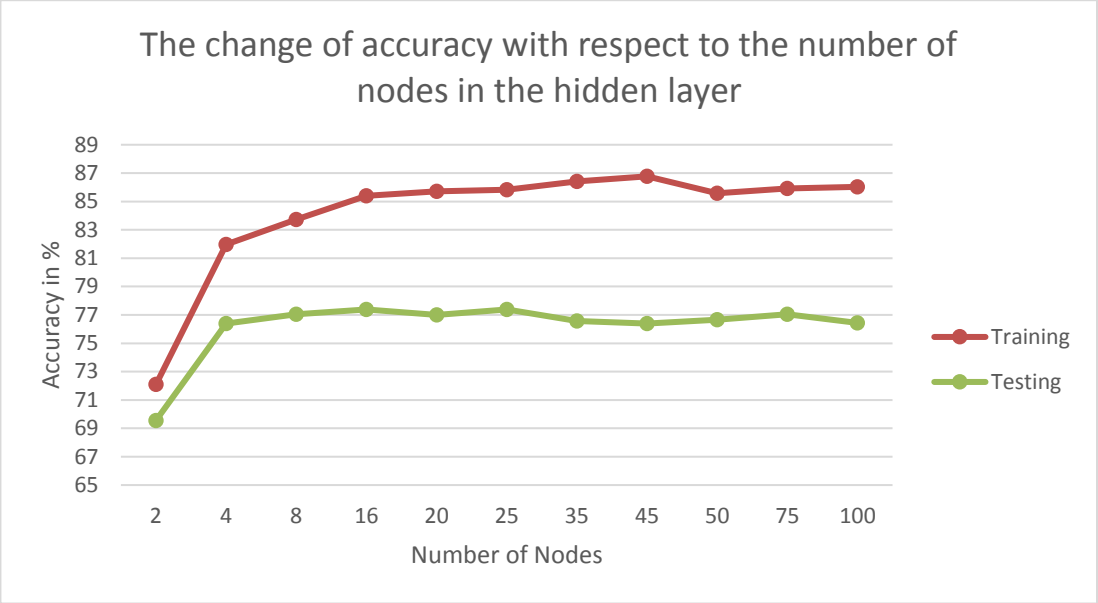

Figure 32 A chart describing how the training and testing accuracy of NN change with the selected number of nodes. Learning rate = 0.05, Hidden Layer = 1.

Figure 32 shows that the effectiveness of backpropagation Neural Networks increases when the number of nodes in the hidden layer increases. We can deduce from the figure that the optimal number of nodes in the hidden layer is 45 when we vary the number 2 to 100.

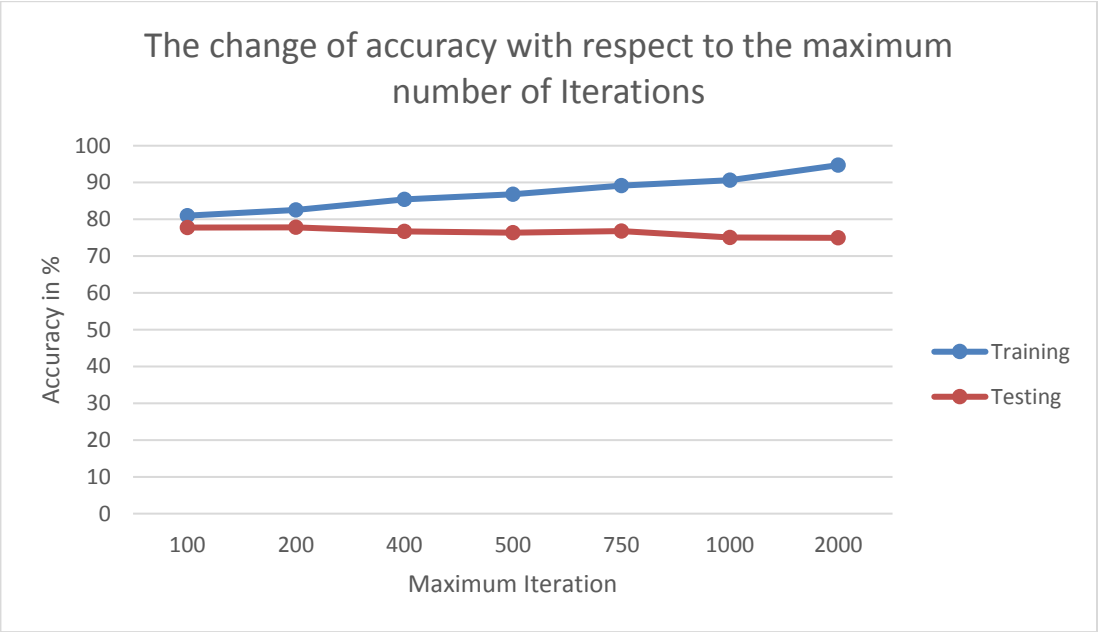

Figure 33 A chart describing how the training and testing accuracy of NN change with the selected maximum number of iterations. Learning rate = 0.05, Hidden Layer = 1, Number of Nodes = 45.

We note from Figure 33 that the testing performance of NN decreases when the number of iterations increases while the training performance increases. This shows us that NN becomes more and more specific to the training data when the number of iterations increases which weakens its performance over new unseen data. The final combination of parameters used is shown in Table 31. These were found to give the best results (reported in Table 30).

Table 31 Optimal parameters to use with Neural Networks

Parameter	Value
Learning Rate	0.05
Number of Hidden Layers	1
Number of hidden Nodes	45
Maximum number of Iterations	750

ANN shows no data overfitting and the technique is stable (over the 10 different folds) as indicated by the low standard deviation. It is important to point out that Neural Networks ignore all the cases with a missing value. Hence, the experiment is done using the original data.

6.2.3 K-Nearest Neighbors

We used the default similarity measurement of KNN - the *Euclidean Distance*. *K* is the main parameter to consider when using KNN. When increasing *K*, we are increasing the number of cases used to classify a new one. This might seem as an advantage, as the classification will be based on a larger number of similar cases than when using a smaller *K*. However, this advantage is dependent on the data and how big is the effect of similar cases on each other. For example, increasing *K* may deteriorate the performance of KNN when dealing with noisy data, given that the larger the *K*, the more noise[16] is included in the neighborhood.

[16] Noise in the data is seen when similar cases have different class labels.

We used Scikit-Learn release 0.18.1 (http://scikit-learn.org/) - a Python library - for machine learning. We run KNN multiple times with a different value of *K* each time. The behavior of KNN with respect to *K* is shown in Figure 34. We finally choose *K* to be 3 as it gives the best results.

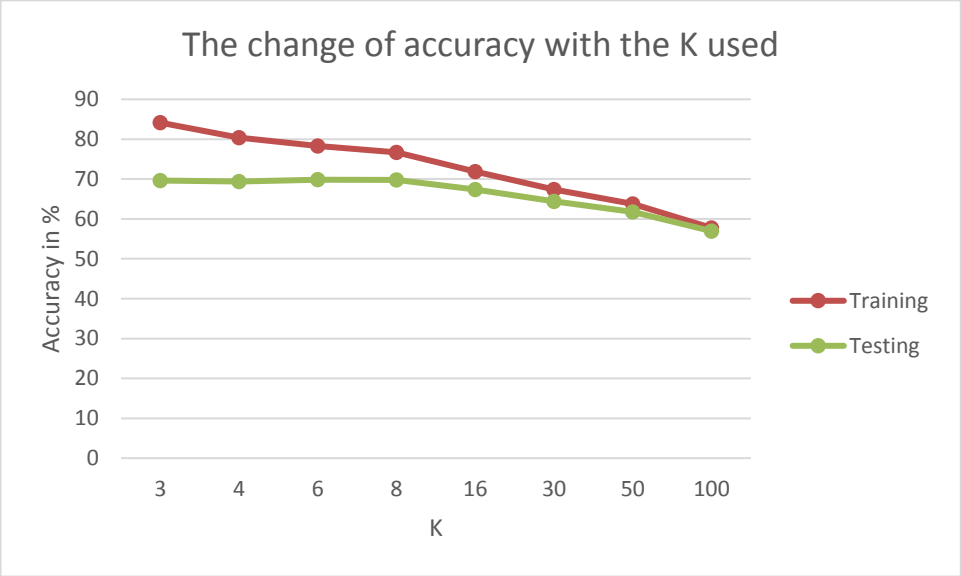

Figure 34 The change of the accuracy of KNN over testing and training data in respect with the k used.

Figure 34 shows a decrease in KNN performance as *K* increases. This is due to noise in the data.

There is no data overfitting and the technique is stable (over the different 10 folds) given the low value of the standard deviation.

6.2.4 Case-Based Reasoning

We implemented the technique using Python. Results show the stability of the technique (over the 10 folds) and that there is no data overfitting (Table 30).

6.2.5 Support Vector Machine

SVM performance is not affected by any parameter. We used the same library as K-NN (Scikit-Learn). There is no data overfitting and the technique is stable (over the different 10 folds) as shown by the low value of the standard deviation.

6.2.6 Genetic Algorithms

Using C5, we generate 50 different rulesets using the training folds. We train the instantiated GA on the same data using the 50 generated rulesets as initial population, then we test the final ruleset produced by GA on the testing fold. We repeated this experiment 30 times because of the random element in GA.

6.2.6.1 The Effect of Parameters

The GA performance is highly affected by 5 parameters: *K* used in the *K*-tournament selection, the probability of mutating the child (μ_1), the probability of mutating the best parent (μ_2), the probability of choosing the rule in the crossover based on the confidence (*Pconfidence*), the percentage of elitism (*ELIT_PERCENT*) and the maximum length of the plateau (*MAXPLAT*). We run GA with different values of the parameters. In Table 32, we report those that gave us the best results on the testing data.

Table 32 The best values of GA parameters

Parameter	Value
K	3
μ_1	0.05
μ_2	0.1
Pconfidence	0.7
MAXPLAT	5
ELIT_PERCENT	20%
Population size	50

6.3 Discussion of Results

As shown in Table 30, Figure 29 and Figure 30, GA outperforms all the other machine learning techniques on the testing data obtaining a testing accuracy of 90.5%, followed by C5 (85.3%). This shows that decision rules are good at prediction from this particular data.

CBR reaches results very close to C5. In contrast to GA, *K*-NN shows the worst results of all. The performance of CBR gives us an insight about the poor performance of *K*-NN, as we can deduce that the Euclidean distance between cases which is the default distance measure used by *K*-NN is not a suitable approach to define similar instances for the problem in hand.

In order to test how significant the improvement of the GA over C5 is, we used the paired t-test (Shier, 2004) which analyses the probability that the difference in performance between GA and C5 is due to chance. The reader can refer to Appendix A for a description of this test. The result of this test shows that improvement of GA over C5 is extremely significant with an approximate p-value of 0.00001 signifying that the improvement of GA over C5 is not due to chance.

6.4 Performance

The running time of our GA is significantly affected by the size of the population, the number of generations and the probability for the mutation. Our GA takes approximately 1,350 seconds to find the optimal ruleset on one fold, when running it on an Intel(R) Xeon(R) E7-8870 v2 @ 2.30GHz CPU with 30720 KB cache and 629GB RAM running Red Hat CentOS Linux, Version 7. The performance can be improved if we parallelize the computation of the fitness function.

6.5 Validation Set

In order to validate our results, we tested GA against the best among the other techniques, namely, C5. We did the test on a new unseen data set consisting of movies from 1998 to 1999 inclusively. As seen in Table 33, GA also outperforms C5 and achieved similar results to the original data set.

Table 33 Performance of GA and C5 on the validation set

Technique	Accuracy (std)
GA	88.16 (1.5)
C5	77.29 (2.5)

6.6 Attribute Effect

The winnowing technique provided by C5 can be used as a method to analyze the attributes correlation with the class label. By this technique, C5 lists the attributes that are eliminated from the decision tree, i.e. the attributes that are ineffective in the classification process. We noticed however, that including such attributes in our GA, gave better results. As a matter of fact, we ran our GA with and without these attributes. In the latter case, the accuracy was deteriorated.

In order to assess attributes effectiveness, we examine the final ruleset produced by GA and keep note of unused attributes. Also, we explore which attributes can be removed from the classifier without deteriorating the accuracy. For this, and for each attribute, we remove all the conditions corresponding to it from the GA optimal ruleset, and if this does not deteriorate the performance (in term of training accuracy), we remove the attribute from the classifier.

In addition, we measure the effectiveness level of each attribute by calculating the decrease in accuracy when removing this attribute. For example, suppose the initial accuracy was 95%, and after removing attribute A, we attain an accuracy of 92%. Thus the effectiveness level of A is 3. Hence, an attribute with a 0 effectiveness level or less can be removed from the study without affecting the performance. The effectiveness level of all the attributes are shown in Figure 35.

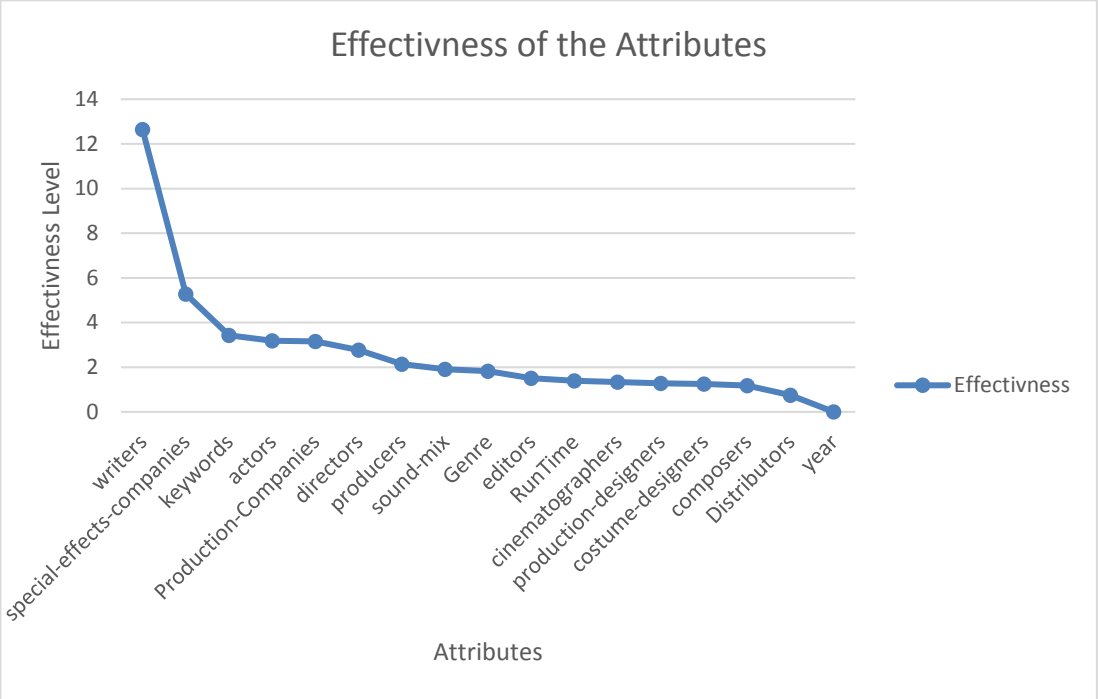

Figure 35 The computed effectiveness level of each attribute

As one can see in Figure 35, we can safely remove the attribute "*Year*" without affecting the performance of our model since it has an effectiveness value of 0.

6.7 Rating and Box Office Growth

As mentioned in Chapter 1, investors and production companies are interested in a movie profit as well as its rating. In Figure 36, we plot the IMDB rating against the box office growth. The graph shows that the box office growth of a movie is highly

correlated with its IMDB rating. Hence, the latter can be used as an indicator of the former.

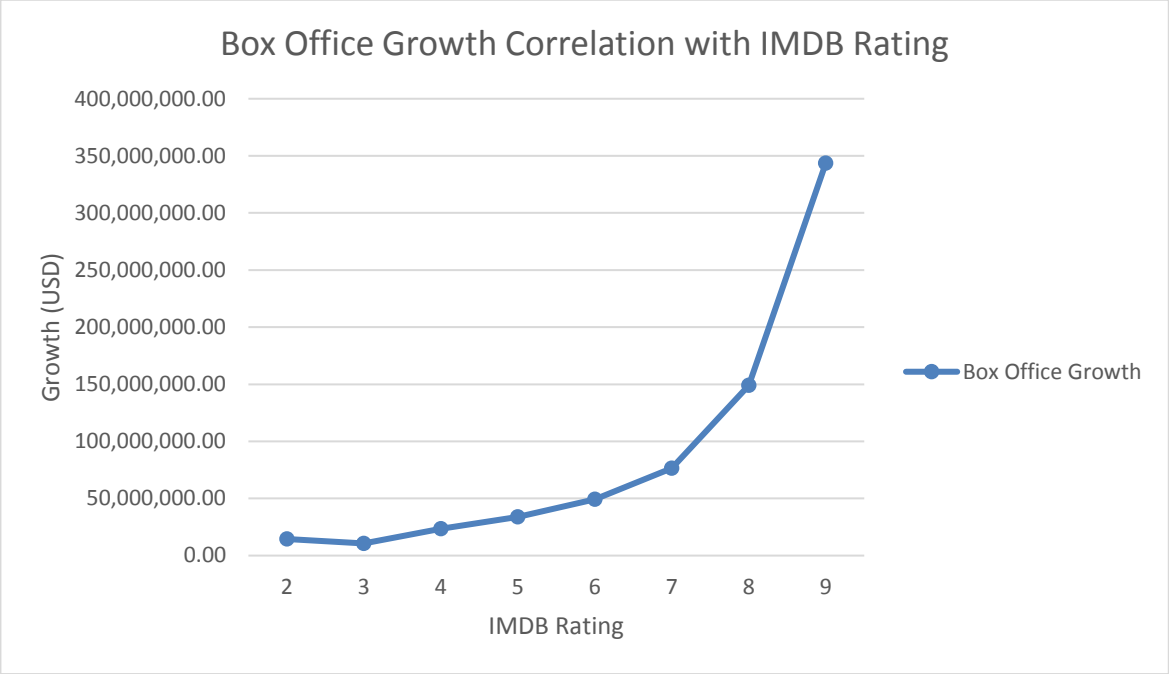

Figure 36 Correlation between IMDB rating and the box office growth

Chapter 7

Conclusion and Future Work

Movie production is a precarious investment field which may result in vast profit or loss. Developing a practical model which predicts the success of movies before their production with a relatively high precision may avert investors and several companies from incurring losses and bankruptcies. In this thesis, we tackle this problem using machine learning techniques namely: Genetic Algorithms, C5, Support Vectors Machine, *K*-Nearest Neighbors, Neural Networks and Case-based reasoning. We tested all the techniques on movies collected from IMDB. These are long U.S movies produced between the years 2000 and 2017. The data set consists of 4883 movies. All techniques gave promising results showing the accuracy of machine learning on this problem. We developed our own genetic algorithm that optimizes rulesets produced by C5. Our implemented GA achieved the highest results with approximately a 90.5% prediction accuracy on unseen movies (88.16% on a validation set of movies from 1998 to 1999).

The advantage of our approach is that it provides a set of rules in a human readable format that help investors and production companies decide on some movie features which raise the success chance of a movie.

We believe our approach can prevent production companies from incurring huge losses as happened with FOX studio and Square Picture in "*Final Fantasy: The Spirits Within (2001)"* and "*Titan A.E (2000)"* respectively.

Our experiments showed that some features affect more the success of movies, precisely, *Writers* contribute the most followed by *Special Effect Companies*, *Keywords*, *Actors*, etc.

In addition, we showed that IMDB rating can be a good indicator of movie success given its high correlation with the movie box office growth.

In future work, we plan to test the performance of other meta-heuristics while using the box office growth as a measure of success.

Also, an interesting idea would be to perform a domain adaptation of our model in order to use it on TV series, documentaries and shows.

Bibliography

Arundeep, K., & AP, N. (2013). Predicting Movie Success Using Neural Networks. *International Journal Of Science And Researche, 2*(9), 69-71.

Asad, K. I., Ahmed, T., & Saiedur Rahman, M. (2012). Movie popularity classification based on inherent movie attributes using C4.5, PART and correlation coefficient. *2012 International Conference On Informatics, Electronics & Vision (ICIEV)*, (pp. 747-752).

Barman, D., Chowdhury, N., & Singha, R. (2012). To predict possible profit/loss of a movie to be launched using MLP with back-propagation learning. *2012 International Conference On Communications, Devices And Intelligent Systems (CODIS)*, (pp. 322-325).

Bhave, A., Kulkarni, H., Biramane, V., & Kosamkar, P. (2015). Role of different factors in predicting movie success. *International Conference On Pervasive Computing (ICPC)*, (pp. 1-4).

Borga, D., & Hasbrouck, R. (2012). WHEN TO GREENLIGHT: Examining the Pre-release Factors that Determine Future Box Office Success of a Movie in the United States. *International Journal Of Economics And Management Sciences, 3*(2), 35-42.

Breiman, L., Friedman, J., Stones, C. .., & Olshen, R. A. (1984). *Classification and regression trees.* Boca Raton, Florida: CRC Press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 3*(20), 293-297.

Driver, M. (2016, July 7). *A Look Back at the Folly of 'Final Fantasy: The Spirits Within'.* Retrieved December 24, 2016, from Vice: www.vice.com

Gammerman, A., Vovk, V., & Vapnik, V. (1998). Learning by transduction. *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (UAI'98)*, (pp. 148-155).

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning, 3*(2), 95-99.

Hand, L. N., & Finch, J. D. (1998). *Analytical mechanics.* Cambridge University Press.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* Ann Arbor: University of Michigan Press.

Hsu, P., Shen, Y., & Xie, X. (2014). Predicting Movies User Ratings with Imdb Attributes. *Rough Sets And Knowledge Technology*(8818), 444-453.

Kawashima, T., Ogawa, T., & Haseyama, M. (2013). A rating prediction method for e-commerce application using ordinal regression based on LDA with multi-modal features. *2013 IEEE 2Nd Global Conference On Consumer Electronics (GCCE)*, (pp. 260-261).

Lash, M. T., & Zhao, K. (2016). Early Predictions of Movie Success: the Who, What, and When of Profitability. *Journal of Management Information Systems, 3*(33), 874-903.

Latif, M., & Afzal, F. (2016). Prediction of Movies popularity Using Machine Learning Techniques. *IJCSNS International Journal of Computer Science and Network Security, 8*(16), 127-131.

Mestyán, M., Yasseri, T., & Kertész, J. (2013). Early Prediction of Movie Box Office Success Based on Wikipedia Activity Big Data. *Plos ONE, 8*(8), e71226.

Miller, B., & Goldberg, D. (1995). Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems*(9), 193–212.

Mitchell, M. (1998). *An introduction to genetic algorithms.* MIT Press.

Mitchell, T. M. (1997). *Machine Learning.* New York: McGraw-Hill.

Nithin, V., Pranav, M., Sarath Babu, P., & Lijiya, A. (2014). Predicting Movie Success Based on IMDB Data. *International Journal Of Data Mining Techniques And Applications*(3), 365-368.

Oghina, A., Breuss, M., Tsagkias, M., & de Rijke, M. (2012). Predicting IMDB Movie Ratings Using Social Media. *Lecture Notes In Computer Science*(7224), 503-507.

Palmeri, C. (2013, September 19). *Despicable Me 2 Producer Knows How to Win the Box Office.* Retrieved December 24, 2016, from Bloomberg.

Parimi, R., & Caragea, D. (2013). Pre-release Box-Office Success Prediction for Motion Pictures. *Machine Learning And Data Mining In Pattern Recognition*(7988), 571-585.

Quinlan, J. R. (1986). Induction of Decision Trees. *Mach. Learn., 1*(1), 81-106.

Quinlan, J. R. (1993). *C4. 5: Programming for machine learning.* Morgan Kaufmann.

Saraee, M., White, S., & Eccleston, J. (2004). A Data Mining Approach To Analysis And Prediction Of Movie Ratings. *WIT Transactions On Information And Communication Technologies*(33), 15-17.

Saranya, A., & Hussain, A. (2015). User genre Movie Recommendation System using NB tree. *International Journal Of Innovative Research In Science, Engineering And Technology, 7*(4), 5854-5859.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 1*(34), 1-4.

Sharda, R., & Delen, D. (2006). Predicting box-office success of motion pictures with neural networks. *Expert Systems With Applications, 2*(30), 243-254.

Shier, R. (2004). Statistics: Paired t-tests. *Mathematic Learning Support Centre.*

Shraddha, M., Hitarthi, B., & Darshana, D. (2015). A Compendium for Prediction of Success of a Movie Based Upon Different Factors. *International Journal Of Advanced Research In Computer And Communication Engineering, 12*(4), 297-300.

Tomar, R., & Verma, C. (2015). User propensity analysis for Movie prediction rating based on Collaborative filtering and Fuzzy system. *International Journal Of Innovative Science, Engineering & Technology, 9*(2), 471-479.

Wang, F., Cai, R., & Huang, M. (2010). Forecasting Movie-Going Behavior Based on Online Pre-Release WOM and Opening Strength. *2010 2Nd International Workshop On Intelligent Systems And Applications*, (pp. 1-4).

Zhang, L., Luo, J., & Yang, S. (2009). Forecasting box office revenue of movies with BP neural network. *Expert Systems With Applications, 3*(36), 6580-6587.

Appendix A

Paired t-test

The paired t-test works by computing the difference between the performance of GA and C5 for each fold. Then, it calculates the average difference over all the folds along with the standard deviation. Table 34 shows the difference of performance (in term of testing accuracy) between C5 and GA.

Next, the paired test compute the average *Standard Error* which is given by Eq. 28

$$Standard\ Error = \frac{Standard\ Diviation}{\sqrt{Number\ of\ folds}} \qquad \text{Eq. 28}$$

The final step is to calculate the T-Statistic value using Eq. 29

$$T\text{-}Statistic = \frac{Average\ Difference}{Standard\ Error} \qquad \text{Eq. 29}$$

Looking up the value of *T-Statistic* to $t_{\text{Number of folds -1}}$ in the t-distribution tables (http://www.sjsu.edu/faculty/gerstman/StatPrimer/t-table.pdf), we can extract the p-value which corresponds to the desired probability.

Table 34 The difference of testing performance between C5 and GA on each fold

Fold	C5	GA	Difference
1	81.2	90.14374	8.943737
2	87.1	92.22222	5.122222
3	84.6	90.2454	5.645399
4	85.7	89.41377	3.71377
5	84	90.51913	6.519126
6	85.5	91.72131	6.221311
7	86.5	90.23224	3.73224
8	84.2	90.2459	6.045902
9	86.1	89.90437	3.804372
10	87.7	90.09563	2.395628

The average difference is 5.214371 and the standard deviation is 1.785837.

Using Eq.28, we can calculate the average *Standard Error* which is:

$$\textit{Standard Error} = \frac{1.785837}{\sqrt{10}} = 0.564731167$$

And The *T-Statistic* is:

$$\textit{T-Statistic} = \frac{5.214371}{0.564731167} = 9.233368$$

Using the t-distribution table, a *T-Statistic* of 9.233368 with a distribution of t_9, we obtain a p-value equal to 0.00001 indicating that the improvement of GA over C5 is extremely significant and not due to chance.

Appendix B

GA Performance Details

B.1 GA Learning Curve

Figure 37 shows the learning curve of GA over the 10 folds. The learning curve shows how and when GA converges to the optimum. As one can, GA converges early, after an average of 10 generations approximately.

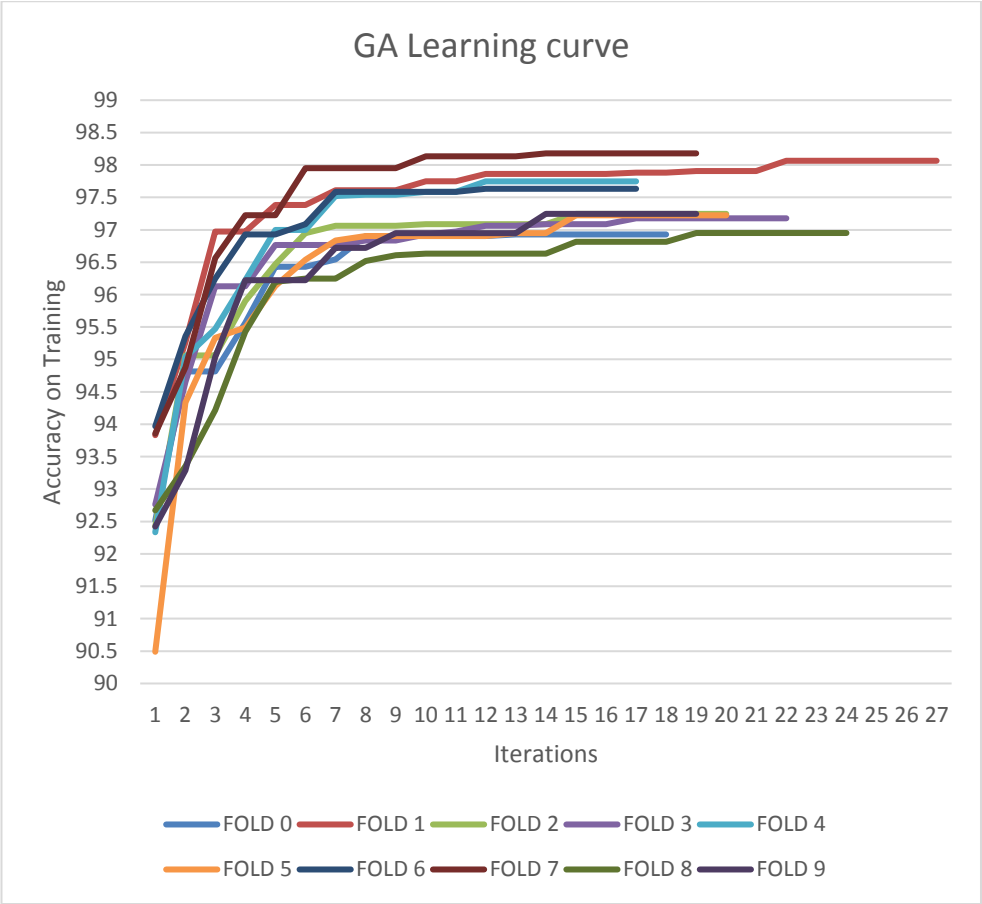

Figure 37 The learning curve of GA

B.2 GA performance per Class Label

As shown in Figure 38, GA performs well on all the rating labels despite the unbalanced nature of our data set.

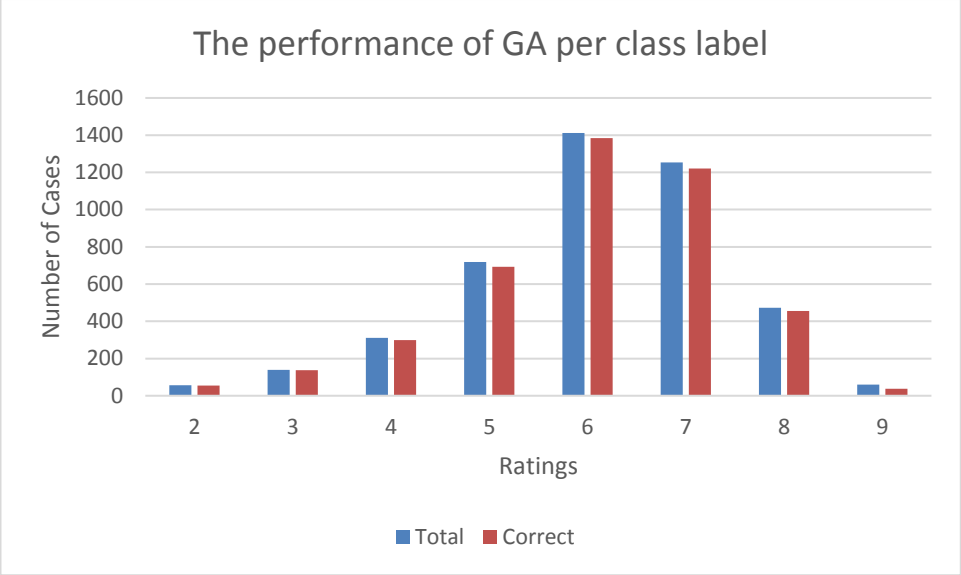

Figure 38 Number of correctly classified cases by GA per class Label (Rating).