Analysis of Success Factors of Movies Based on Internet Movie Database

Thesis submitted for the Master's degree
in computer science and engineering
by Nicolas Pirotte and Hervé Eerebout

Supervisors: Yves Deville and Michel Verleysen
Reader: Cyrille Dejemeppe

Louvain-la-Neuve
Academic year 2014-2015

Contents

1 Introduction **3**

2 Theory of Models **5**
2.1 k-Nearest Neighbors Algorithm 5
2.1.1 Distance 6
2.1.2 Standardisation 7
2.1.3 Average of the k-Nearest Neighbors 8
2.1.4 Cross Validation 8
2.2 Linear Regression 9
2.2.1 Estimation with Ordinary Least Squares Method . . . 11
2.3 Polynomial Regression 12
2.4 k-Means Clustering 14

3 Process **16**
3.1 Dataset 16
3.2 Arithmetic Average as Prediction 19
3.3 k-Nearest Neighbors 20
3.4 Linear Regression 22
3.5 k-NN Algorithm with Coefficients from Linear Regression . . . 26
3.6 Quadratic Regression 28
3.7 k-NN Algorithm with Coefficients from Quadratic Regression . 30
3.8 k-Means Clustering 30
3.9 Distance-Weighted k-NN Algorithm 32

4 Result **34**
4.1 Test Phase 34
4.2 Result Phase 35
4.3 Statistical Hypothesis Testing 38
4.4 Algorithm Tuning 40
4.4.1 Drop Features 40
4.4.2 Number of Clusters 42
4.5 Best technique 44
4.6 Implementations 46
4.7 Prediction of a future movie 47

5 Conclusion **49**

Abstract

Forecast the success of a movie before its release can be very interesting for the film industry but it is difficult to achieve. One technique is to use instance-based algorithms to predict the movie ratings based on a dataset about existing movies. In this thesis, we apply different machine learning methods based on a dataset from the website Internet Movie Database (IMDb) to analyse the success factors of movies in order to predict the movie ratings. The different methods used are the k-nearest neighbors algorithm, the linear regression, the quadratic regression and the k-means clustering. Thanks to these methods and the huge dataset from IMDb, we developed interesting machine learning techniques to foresee the movie ratings with a good accuracy. The techniques used in this thesis could be applied in the film industry to guide the director of a movie to make choices about some features of its movie.

1 Introduction

It can be hard to foresee if a movie would be a success. Indeed, a movie can be characterised by a lot of different factors and it's not always obvious to know how these factors would have an impact on the success of a movie.

The success of a movie can be perceived in many ways. The success could mean lots of entrances, or good critics ratings and so on. In this thesis, we have made the choice that the success of a movie is measure by the ratings given by the people. Thus, a movie has more success than another if it gets better ratings.

This thesis analyses in depth the success factors of movies to attempt to predict the movie ratings. To be able to predict movies ratings, we need to have a dataset. The dataset contains lots of movies and each movie is defined according to different features. We will use the Internet Movie Database to construct a consistent database.

Internet Movie Database is a well-known website for movie enthusiasts. As it says on the official website, IMDb started in 1990 as a hobby project by an international group of movie and TV fans.
IMDb is now the world's most popular and authoritative source for movie, TV and celebrity content. The website offers a searchable database of more than 100 million data items including more than 2 million movie, TV and

entertainment programs and more than 4 million cast and crew members.

Thanks to the IMDb website, we were able to retrieve all the data we needed to characterise each movie. As we explain deeper in the next section, we got around sixty thousand movies. IMDb contains more than sixty thousand movies but every movie doesn't have all the features required to characterise a movie.

The goal of this thesis is to predict as accurately as possible the users rating of movies which aren't released yet. And at the same time, we try to define the factors which are relevant for a movie such as the director name, the writer name, and so on. Actually, these two goals are correlated because we need to determine the key factors of movies to compute the prediction of the user ratings of a movie.

The thesis is separated in four sections. The first section explains all the theory of the models we have used. Along this thesis, we used several different theory models. These models have to be understood by every reader before reading the process section.

Then, we carry on with the process section. This section explains how we fit and mix the models to our case. It details every step we have followed to get results. Each next subsection is a consequence of the previous subsection. You will clearly observe the reasoning behind each step.

After this, we have a result section. It will be useful for the analysis of all the results we got. We will observe each result and understand the meaning of it. In this section, we also explain how we implemented our programs. Finally, we will conclude with a summary.

2 Theory of Models

This section describes all the different models and techniques applied in this thesis to achieve the goal of predicting the movie ratings based on a dataset. The first three models which are the k-NN regression, the linear regression and the quadratic regression are regression methods whose goal is to predict the movie ratings while the last one, the k-means clustering, is an algorithm to partition the dataset into different clusters.

This section contains just the theory of the models, their application in the scope of this thesis is explained in Section 3 and all the results of the experiments are presented in Section 4.

2.1 k-Nearest Neighbors Algorithm

The k-Nearest Neighbors algorithm [1], also known as k-NN algorithm, is a very popular algorithm in machine learning used for classification and regression. In classification, the algorithm defines the class membership of a new instance among a dataset while the regression estimates a value with the average of its neighbors.

This algorithm belongs to the family of instance-based learning (or memory-based learning) and, therefore, to the lazy learning. The goal of this kind of algorithm is to compare the new input instance directly to a range of training set, stored in memory, as opposed to the eager learning trying to generalise the problem before querying it with a formula like the linear regression (Section 2.2).

A major consequence of these algorithms is the linear complexity on the training set for each instance to predict because there is no learning in this algorithm. Indeed, the complexity grows with the amount of data of the training set to reach a linear complexity O(n). Moreover, it implies to store all the training dataset in memory which can represent a large space requirement.

In the case of this thesis, we use the k-NN regression to predict a continuous variable [1] corresponding to the movie ratings. For this reason we will focus only on the k-NN regression. The protocol of the k-NN regression is quite simple and follows the different steps explained below.

2.1.1 Distance

The first step is to find the nearest neighbors of the new instance with a distance function. We can use different measures to compute the distance between the new instance and the training set like the Euclidean, Manhattan or Minkowski distance functions. In our case we use the most popular one, the Euclidean distance function with the formula

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}, \tag{1}$$

where k is the number of coordinates, x_i is the coordinate i of an instance from the training set and y_i the corresponding coordinate of the new instance.

For instance, let us assume we try to predict the rating of a new movie based on the year of production and the number of awards already received by the main actor (see Figure 1). In this case, for the first distance, x_1 and y_1 correspond respectively to the year of the first example from the training set (1993) and the new instance (2015) and x_2 and y_2 correspond respectively to the awards received by the main actor of the first instance of the dataset (1) and the number of awards received by the main actor of the new movie (1). Therefore, the computation of the first distance is

$$D_1 = \sqrt{(1993 - 2015)^2 + (1 - 1)^2} = 484. \tag{2}$$

Year	Award Actor	Rating	Distance
1993	1	6.8	484
1998	0	6.4	290
1999	1	7.1	256
2005	2	7.6	101
2010	0	5.9	26
2015	1	???	

Figure 1: Euclidean distance of movies

2.1.2 Standardisation

We can notice in the previous example that the year of the movies has a huge impact on the distance compared to the award coordinate because the difference between the years is more important. Therefore, the result is distorted because of the different measurement scales of the coordinates. We can solve this problem by standardising the data in each column to obtain a value between 0 and 1 in order to have all the coordinates with the same influence in the computation of the distance. In this case, the formula applied to standardise the data is

$$X_s = \frac{X_O - X_{min}}{X_{Max} - X_{min}}, \tag{3}$$

where X_s is the standardised feature, X_O is the original feature, X_{min} and X_{Max} are respectively the minimum and the maximum value of all the values of that feature.

This formula is the difference between the current value with the minimum of the column divided by the difference between the maximum and the minimum of the column. Then, we can recompute the new distance values with the standardised data. The new table with the standardised values is shown in Figure 2.

As you can observe, we have to deal with a special case. If $X_{min} = X_{Max}$, the denominator will be equal to zero and the division will be impossible. Therefore, the feature will be discarded. It means that the feature won't have any impact on the decision process of the closest neighbors (Section 2.1.3).

Year	Award Actor	Rating	Distance
0	0.5	6.8	1
0.23	0	6.4	0.92
0.27	0.5	7.1	0.73
0.55	1	7.6	0.67
0.77	0	5.9	0.55
1	0.5	???	

Figure 2: Euclidean distance of movies with standardised data

Now that we have a fair distance between all the features, we can start the last step of the algorithm.

2.1.3 Average of the k-Nearest Neighbors

Thanks to the standardisation and the computation of the distance, we are now able to find the nearest neighbors of the new instance. The k value from k-NN represents the number of neighbors we are going to consider to provide the prediction. The neighbors are selected by increasing distance (the closest neighbor) which are instances closer to the new instance.

For a given target feature, the prediction is based on a simple arithmetic average function of its k nearest neighbors. We can also use the distance-weighted k-NN algorithm [2] to weight the contribution of the neighbors in order to raise the influence of the nearest neighbors by giving a weight of 1/d to each neighbor, where d is the distance value.

Considering the previous example, if you have k = 3, the rating of the new movie without weight is

$$\frac{7.1 + 7.6 + 5.9}{3} = 6.87 \tag{4}$$

and the same example with the weight of the neighbors equal to 1/d is

$$\frac{\frac{7.1}{0.73} + \frac{7.6}{0.67} + \frac{5.9}{0.55}}{\frac{1}{0.73} + \frac{1}{0.67} + \frac{1}{0.55}} = 6.70. \tag{5}$$

After these three steps, we are able to predict the target feature of a new instance based on the training dataset. The last challenge is to find the optimal number k of nearest neighbors. This value can be found using cross validation, a method based on mean squared error (MSE).

2.1.4 Cross Validation

Cross validation [1] is a technique used in statistics to estimate the validity of a model based on sampling. This technique can be used to find the optimal k for the k-NN algorithm. There are several techniques of cross validation like the holdout method or the k-fold cross-validation among the most popular ones.

The holdout method consists in splitting the dataset in a training set (more than 60% of the dataset) and a test set and applying the k-NN algorithm with different values of k. The optimal value of k is the one giving the minimum deviation based on MSE. The mean squared error [3] is used to measure the difference between a predicted value and the real value actually observed with the formula

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2, \tag{6}$$

where n is the number of instance predicted, $\hat{Y}$ is the predicted value and Y is the real value observed.

With the k-fold cross-validation technique [2], the dataset is split into k_f samples. We select one of them as the test set and the $k_f - 1$ others as the training set, then we compute the MSE. We iterate this operation k_f times in order to use each sample as the test set. Finally, we compute the average of the k_f MSE to find the deviation. As with the holdout method, the optimal k of the k-NN is the one giving the minimum deviation.

The following figure summarises the whole process with the different steps of the k-NN algorithm.

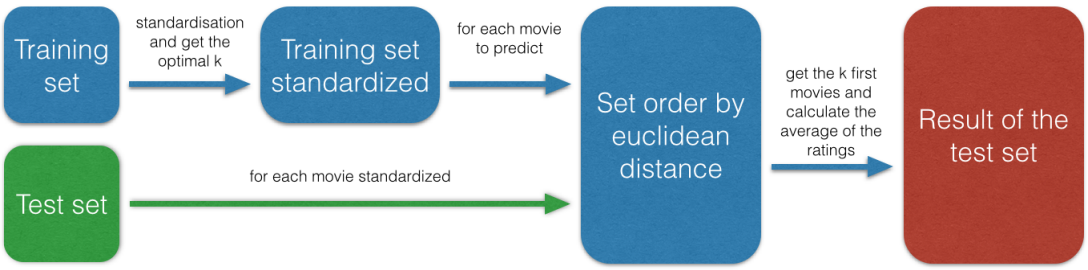

Figure 3: k-NN Process

2.2 Linear Regression

The linear regression [4] is a regression model used in statistic describing the relationship of a scalar dependent variable y with one or more explanatory variables x. The principle is to model the relationship between the dependent variable and the vector containing the explanatory variables with the

assumption that the function associated is linear. Therefore, the function can be represented by

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + u, \tag{7}$$

where y is the dependent variable, the variables x are the explanatory variables, k is the number of explanatory variables and u is the error term. Finally, the variables β are the parameters of the model to estimate. This function can be used to predict numeric value as explained in [3]. Or it can be represented in a vector form as

$$y = \mathbf{x}^T \boldsymbol{\beta} + u, \tag{8}$$

where ^{T} denotes the transpose, $\mathbf{x}$ and $\boldsymbol{\beta}$ are the vectors

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}, \ \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}. \tag{9}$$

The variables β_k and x_k are the same variables as in the Equation 7.

If there is just one explanatory variable x the model is called simple linear regression and can be represented by a 2D-graph, as shown in Figure 4. The dots are the observed values linking the dependent variable y with the explanatory variable x and the line is the linear function estimating the model.

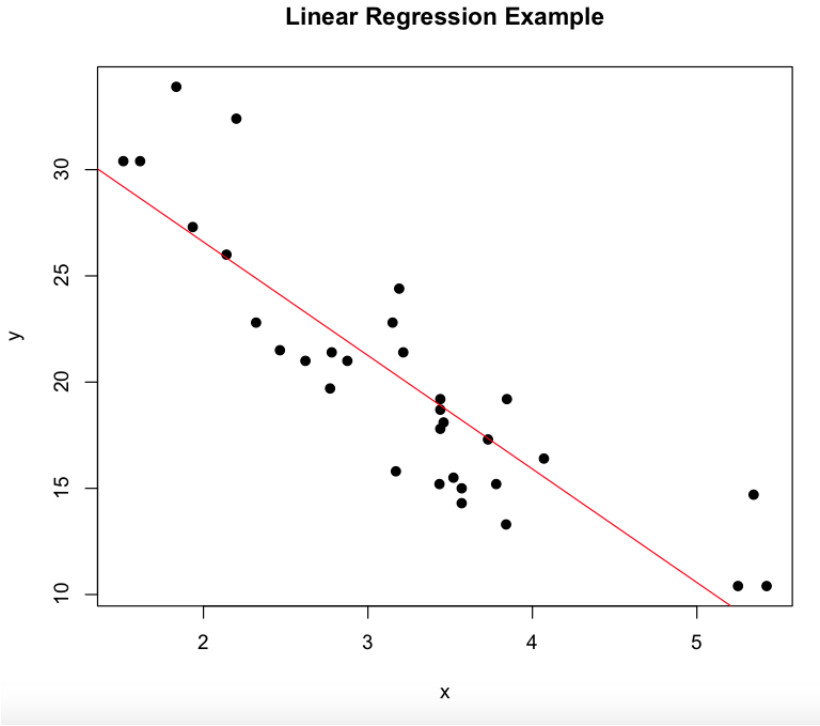

Figure 4: Simple linear regression

Otherwise, if there are more than one explanatory variable, the model is called multiple linear regression.

2.2.1 Estimation with Ordinary Least Squares Method

The most common method to estimate the parameters of a linear regression model is the method of ordinary least squares (OLS) or linear least squares. The goal of this method is to minimise the difference between the real observed values and their predicted values with the linear approximation, also known as residual.

We can express the error $\hat{\epsilon}$ with the difference between the real value y and the estimation $\hat{y}$,

$$\hat{\epsilon} = y - \hat{y}\,.$$

The linear least squares method computes the sum of the squared residuals and reaches its optimum when this sum is a minimum,

$$\min \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \min_{\beta_0,.,\beta_p} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i,1} - \cdots - \beta_p x_{i,p})^2,$$

where n is the number of instances to predict, $\hat{\epsilon}$ is the error of the prediction, the variables β are the parameters of the model, y is the observed value, p is the number of explanatory variables and the variables x are the explanatory variables.

The squared residuals are used by the OLS in order to eliminate the distinction between the positive and negative residuals and to obtain a representative positive value. Indeed, if the sum of the residuals is used instead, the average would reach 0 and the method would make no sense.

The extreme simplicity of the OLS makes it very popular and frequently used in many different domains. Even though there are a lot of other techniques to predict the parameters of a linear model like the maximum-likelihood estimation [1] or the Bayesian inference [1], OLS is still the most used technique.

2.3 Polynomial Regression

The polynomial regression is a regression method modelling the relationship between the dependent variable y with one or more explanatory variables x as a polynomial of degree n.

A simple polynomial model (containing one independent variable x) of degree n can be represented with the formula

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n + \varepsilon,$$

where y is the dependent variable, x is the independent variable, ε is the error term, n is the degree of the polynomial and the variables β are the parameters of the model to estimate.

The simple polynomial model can be represented by a 2D-graph, as shown in Figure 5, where the dots are the observed values linking the dependent variable y with the explanatory variable x and the curve is its estimation.

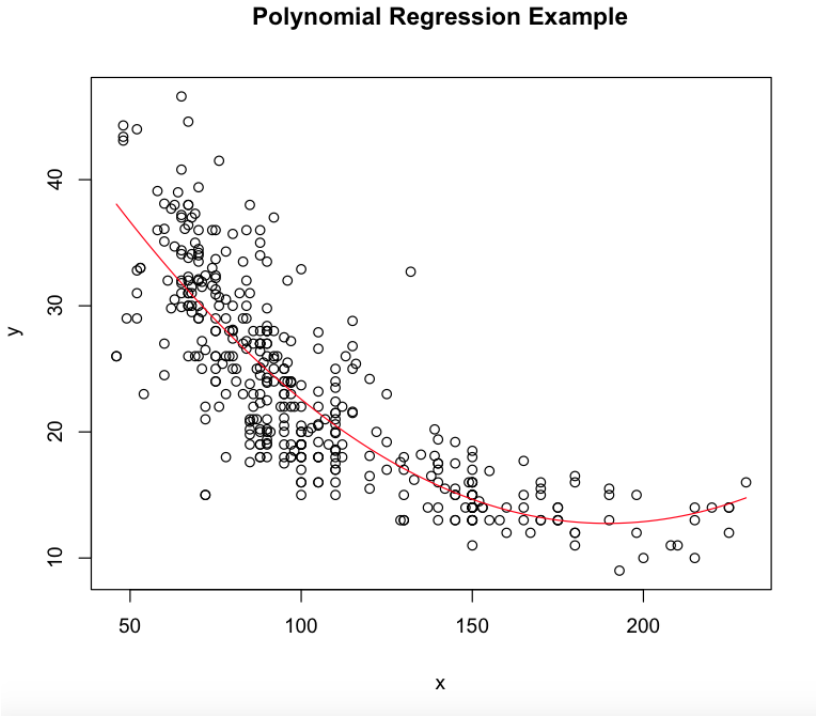

Figure 5: Polynomial regression graph

The polynomial regression is quite similar to the linear regression and can be considered as a special case of the multiple linear regression. Although the model is not linear to the data, the model and the ordinary least squares method consider x_i and x_i^2 as two different variables and don't distinguish that x_i^2 is the square of x_i. Therefore, a simple polynomial regression,

$$y = \beta_0 + \beta_1 x + \beta_2 x^2, \tag{10}$$

can be represented as a multiple linear regression,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2, \tag{11}$$

if $x_2 = x^2$.

The ordinary least squares method is also used to compute the parameters of the polynomial model, like the linear regression. This method is explained in depth in the previous Section 2.2.2.

2.4 k-Means Clustering

The k-Means Clustering [1][4] is a data mining technique used to split a dataset of n observations into k clusters in such a way that the observations belonging to a same cluster are more similar to each other than to those inside the other clusters.

The k-Means Clustering is not a machine learning algorithm whose the aim is to predict a value like the other models presented in this section. However, the clustering give us the possibility to apply other prediction models inside the cluster whose the new instance belongs. This aspect will be described in the explanation of our process in Section 3.8.

To achieve this goal, the algorithm minimises the sum of the Euclidean distance of all the observations with the mean of their cluster,

$$\min \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 , \tag{12}$$

where k is the number of clusters, S represents the set of observations inside a cluster, the variables x are the observations inside the cluster and μ is the mean of the cluster.

The standard clustering algorithm is the iterative distance-based technique [3] following these steps:

1. We set randomly the mean of the k clusters.
2. All the observations are assigned to the cluster with the nearest mean computed thanks to the Euclidean distance function.
3. The mean of each cluster is re-computed with all its observations.

We start again the step 2 and 3 until a state of convergence has been reached.

An example of the execution of the algorithm with a graphical view is shown in Figure 6. The three different colours (red, green and blue) represents the three different clusters. After each iteration, we can see that the means are re-calculated and that some observations change of cluster. The clusters of the fifth and the sixth iteration are similar, which proves that the convergence has been reached.

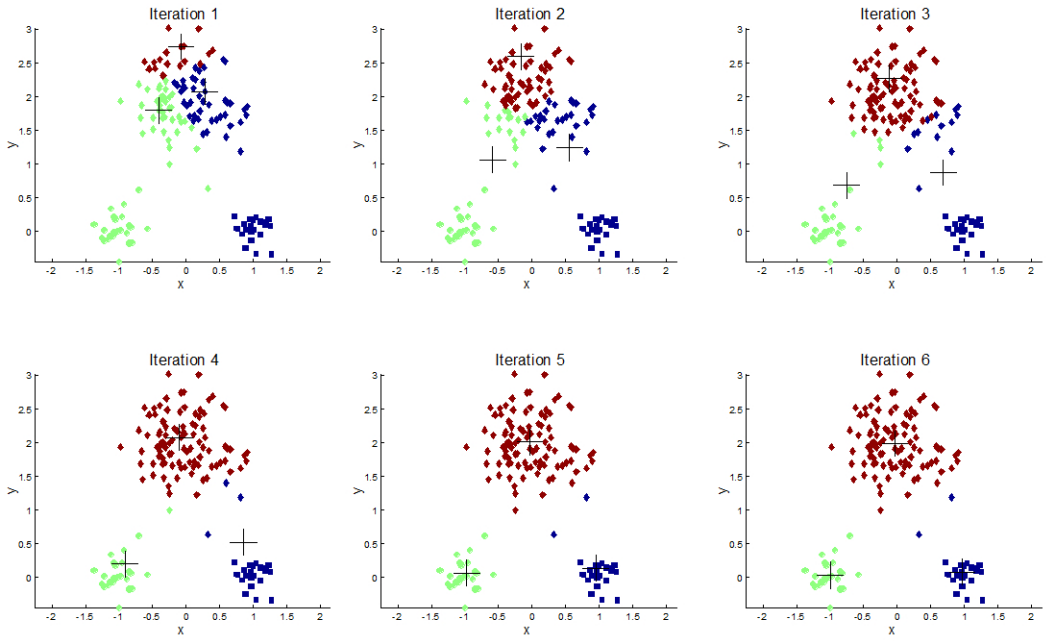

Figure 6: k-Means Clustering example [5]

The convergence state of the algorithm is a local solution but not necessary the optimal solution. Therefore, the first step being random, the results of two successive clustering's iteration are not always the same.

3 Process

The goal of this thesis is to analyse the success factors of movies and to predict their rating by applying machine learning methods. The theory of the machine learning methods used in this thesis are described in Section 1. The learning methods need a dataset to predict a value and, in this case, we collected the data from Internet Movie Database (IMDb).

To apply and evaluate these methods, we split randomly the dataset into a training set (90%) and a test set (10%) such that,

$$training \cap test = \emptyset, \tag{13}$$

the intersection between the training set and the test set is empty. Then, we apply the model to the test set based on the training set to predict the ratings of the movies present in the test set [6]. The performance of a technique is measured with the mean squared error which is used to measure the difference between a predicted value and the real value actually observed for all the movies in the test set (see Equation 6). We implemented several programs to apply these models, the explanation of the implementations are described in Section 4.

This section explains in depth the way we used the different models in order to predict the movie ratings and make this prediction more accurate. Each following subsections represents a model applied and explains the reasons why we used it, the way we applied it and its performance in this context, except the first point presenting the dataset we used with few statistics. This section contains a few results and diagrams to provide a guideline and justify our approach but all the results of the experiments are presented in Section 4.

3.1 Dataset

We collected the data from IMDb to provide the dataset to the learning methods predicting the movie ratings. The rating of a movie on IMDb is the rating given by the users of the website; this value is between 0 and 10. The first step was to collect the data from the website IMDb. Obviously, the data collected could only be numerical values in order to apply regression algorithms on these data. To complete this requirement, we developed a program parsing automatically all the URL's of the IMDb website corresponding to the movies to extract all the features about the movies.

After one week of computation, we got a database of 60.089 movies. Although IMDb contains more movies, we chose to stop the parsing because of the time that involves the download of each HTML file. Moreover, we had to skip a lot of movies that didn't have enough available data on IMDb to be integrated in the database. The list of the mandatory features is shown in Figure 7.

For all the movies in the database there are 37 different features. The list of the collected features is presented in Figure 7. These features are all known before the release of the movie except for the feature to predict, the rating. Indeed, we want to predict the rating of a new movie, therefore, a feature like the number of awards received by the movie can't be used to predict the rating of a new movie but, instead, could be predicted like the rating.

#Movie Actor 1	#Movie Actor 2	#Movie Actor 3
#Production Actor 1	#Production Actor 2	#Production Actor 3
#Direction Actor 1	#Direction Actor 2	#Direction Actor 3
#Writing Actor 1	#Writing Actor 2	#Writing Actor 3
#Award Actor 1	#Award Actor 2	#Award Actor 3
#Nomination Actor 1	#Nomination Actor 2	#Nomination Actor 3
Date of Birth Actor 1	Date of Birth Actor 2	Date of Birth Actor 3
#Movie Director	#Movie Writer	Year of Prod
#Production Director	#Production Writer	RATING
#Direction Director	#Direction Writer	
#Writing Director	#Writing Writer	
#Award Director	#Award Writer	
#Nomination Director	#Nomination Writer	
Date of Birth Director	Date of Birth Writer	

Figure 7: Features of movies

The features include the year of production of the movie, the rating and 7 features about 5 different people: the first, second and third actors, the director and the writer. The 7 features about the people are the number of movie as actor, as producer, as director, as writer, the number of awards received, the number of nominations for these awards and the date of birth of the person. These features are listed in that order in Figure 7.

The dataset has some limitations because some features change over time like the number of nominations or awards received and the number of movies produced, directed or written. Indeed, the dataset is up to date and predicting the rating of older movies can give distorted results. For instance, predicting the rating of a movie of the 90's with the data corresponding to the main actor in 2015 (taking into account all the awards received after the release of the movie) makes no sense.

However, the 37 features were just used at the beginning of the thesis and we reduced this number to 21 by eliminating the less important features. The explanation and the way we rejected them are described in Subsection 3.4 about the linear regression. All the following results and conclusions are based on these 21 features, underlined in Figure 7.

The most important feature is the feature to predict, the rating. An interesting statistic about the ratings is the dispersion diagram of these values shown in Figure 8. We can note that the graph looks like a Gaussian function; this observation will be important to justify some results of the next sections.

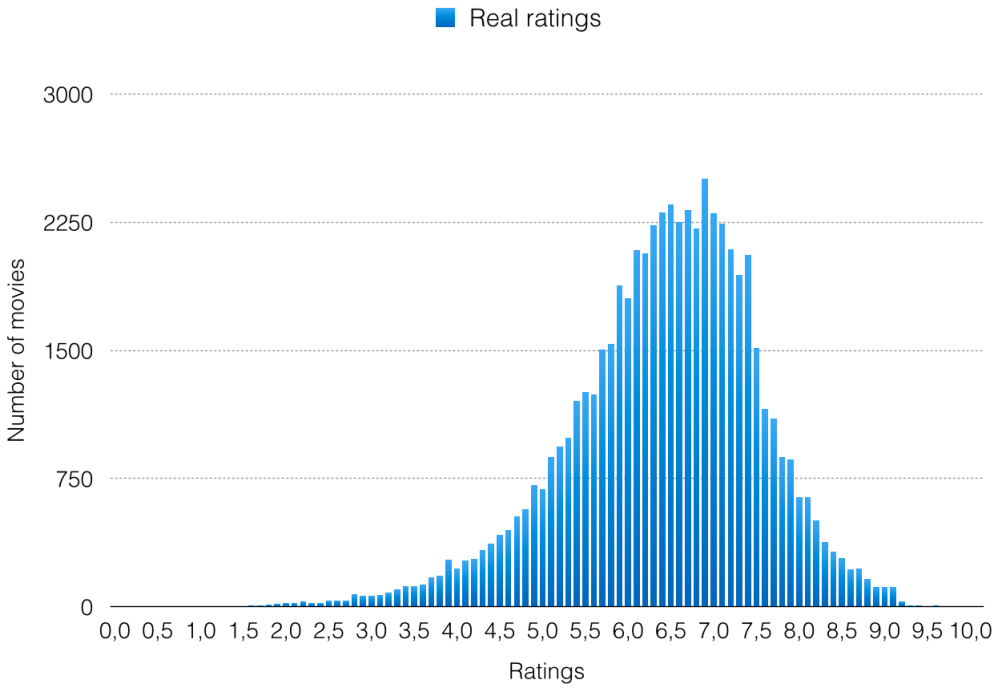

Figure 8: Dispersion Diagram

Each column of this diagram represents, on the horizontal axe, the value of a rating between 0 and 10 by steps of 0.1 as the ratings present on IMDb.

3.2 Arithmetic Average as Prediction

The first technique applied to predict the rating is to give the arithmetic average of the training set as prediction for each movie of the test set. This method is very simple to use and can be very useful. Indeed, at the beginning of the experiments, we don't have any results yet and we need a performance benchmark to evaluate if the other models give good results compared to the basic prediction based on the simple average. So this model is the baseline model.

Moreover, as seen in Section 3.1, the dispersion diagram of the real ratings of the dataset (Figure 8) shows a Gaussian function with a peak in its average, meaning that a big proportion of the ratings is close to the average rating. Therefore, the fact to give the average as prediction for each

instance can actually give good performance to begin with. The MSE, with this model, is equal to 1.25. All the other experiments will attempt to have a MSE lower than 1.25.

3.3 k-Nearest Neighbors

The first machine learning model applied was the k-NN algorithm. Our program follows, of course, the theory of the k-NN as explained in Section 2.1. However, the k-NN algorithm can be implemented by different ways according, for instance, to the distance function used or the kind of average computed. In this thesis, we used the Euclidean distance function (see Equation 1) and, at the beginning, a simple arithmetic average function to compute the ratings predicted.

Moreover, one important choice was which kind of cross validation to use to find the optimal k. In order to limit the computation time, we chose to use the holdout method (see Section 2.1.4) in one way. Therefore, we performed a k-NN inside the training set to find the value of k. Then, we compute the k-NN algorithm on the movies of the test sets with the k value obtained earlier (see Figure 9).

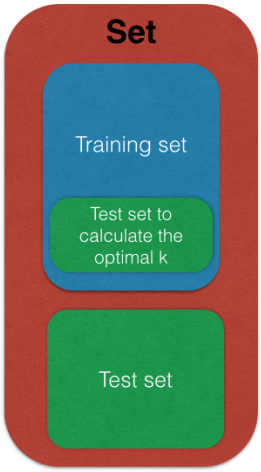

Figure 9: Cross validation

The first results of this algorithm were very promising. Indeed, we improved the MSE of the previous technique based on the arithmetic average

and we got a MSE equals to 1.129.

Figure 10 shows the mean squared error according to the real rating. In this diagram, each column represents the value of a MSE. It is computed between all the instances having the same real rating. The horizontal axis represents the ratings between 0 and 10 while the vertical axis represents the values of the MSE's. We can notice that we can predict with a good accuracy the movies whose rating is close to the average rating because, as seen in the dispersion diagram of Figure 8, most of the ratings are close to this point. Indeed, the nearest neighbors have more chance to have a rating close to the peak and, thus, the average moves towards this value.

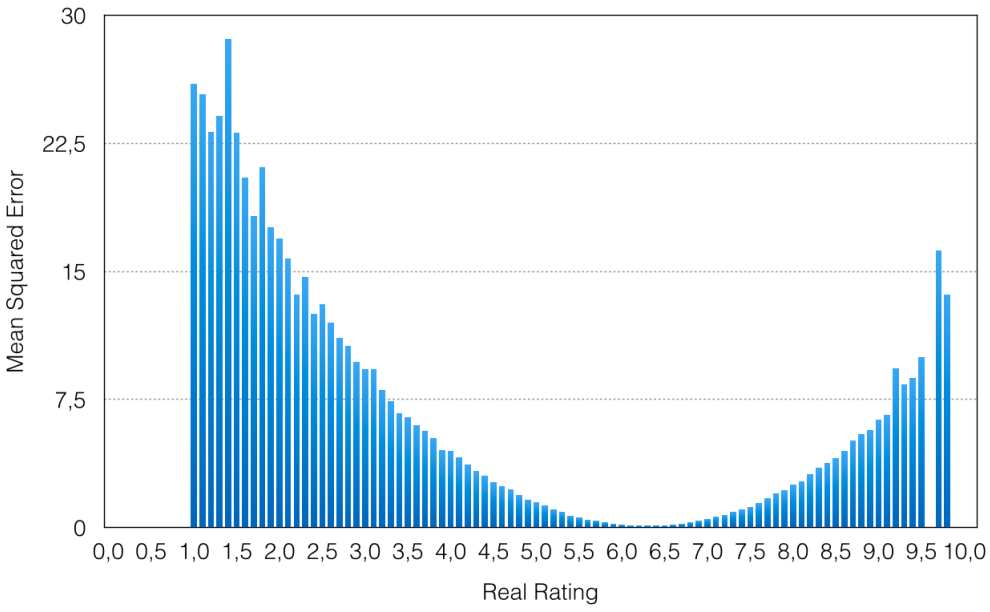

Figure 10: MSE by rating with the k-NN algorithm

Although we got better results with this algorithm, we wanted to improve them. We had the idea to influence the impact of the different features of the movies according to their importance, it's why we started to use the well known linear regression.

3.4 Linear Regression

We used the linear regression model to achieve three important goals. After getting the results with the k-NN algorithm, we tried to enhance these results by using new models. The linear regression helped us in obtaining better results.

As we described in the section about the theory of models, the linear regression model allows to obtain a function depending on the correlation between each parameter. Thus, we wanted to explain the ratings by taking into account the 36 different features about a movie.

The function takes the following form:

$$rating = \beta_0 + \beta_1 f_1 + \beta_2 f_2 + \ldots, \tag{14}$$

where f_1 could be equal to the year of the production of the movie and f_2 the birthday of the first actor.

Thanks to the linear regression, we got the coefficients of the parameters β which can be considered such as the weight of each feature. The β_0 is the intercept and is not linked to a feature.

The first goal we tried to achieve was the prediction of the ratings of a test set by using the function above. Thereby we were able to observe if the MSE of this computation was lower than the simple k-NN algorithm. We also wanted to see the repartition of the results and notice if the prediction of a movie was often around the average rating or not.

To perform the linear regression, we used a software called R to compute the coefficients of each parameters. Thanks to these coefficients, we were able to compute predictions of ratings and to evaluate the performance of this technique with the MSE.
The exact steps to compute these results are detailed in Section 4.

The process of the linear regression model applied in this case is described in Figure 11. First of all, we applied the linear regression on the training set to get the coefficient for each feature and, then, we applied the linear function with the coefficients on the instances from the test set.

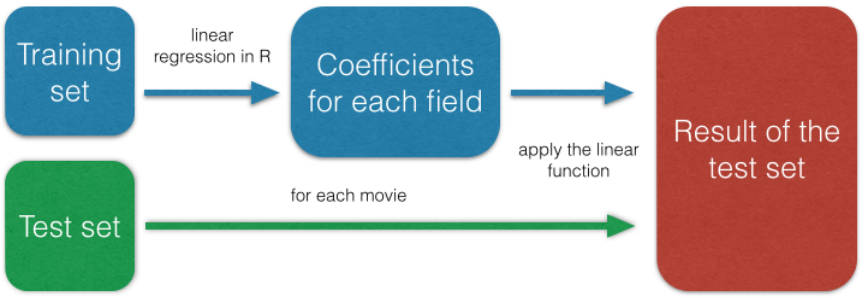

Figure 11: Regression Linear Process

The result of that was a MSE equals to 1.197 which is higher than the previous model. After that we created a graph, as we did for the k-NN, to obtain good representation of the errors of prediction with regards to the real ratings. When we observed the graph in Figure 12, we saw that the function predicted very well the movies which had ratings in the range of 6/10 to 7/10.

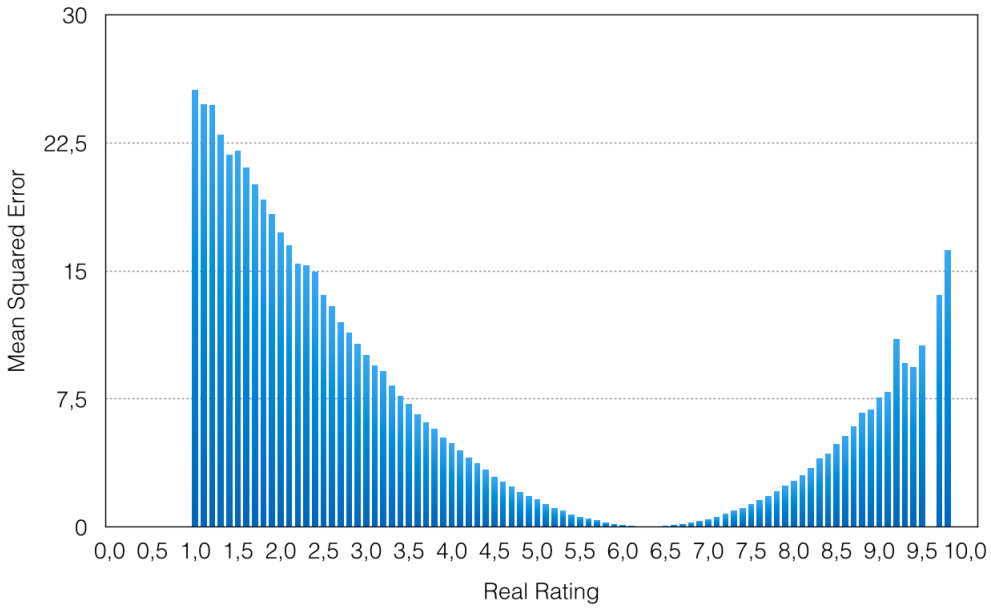

Figure 12: MSE by rating with the linear regression

As a consequence, to obtain a better understanding of this observation, we generated another graph which describes the repartition of the predicted

ratings compared to the repartition of the real ratings. We found out that almost every predicted movies got a rating between 6.5/10 and 7/10 (see Figure 13) while the repartition of the real ratings is more widely distributed. That explains why the function gives only good predictions within this range. We concluded that the function obtained by the linear regression didn't give better prediction than the k-NN method. We also noted that the distribution of the predicted ratings was worst than the k-NN.

As explained above, Figure 13 shows the repartition of the predicted ratings compared to the repartition of the real ratings. Each column of this diagram represents, on the horizontal axis, the value of a rating between 0 and 10 by steps of 0.1 as the ratings present on IMDb. However, the predicted ratings are not scaled to the precision of one decimal digit as the ratings on IMDb but usually have much more precision with more decimal digit. Therefore, to draw this diagram, as the other dispersion diagrams with predicted ratings, we truncated the predicted ratings to one decimal digit to fit them in the diagram. For instance, for a predicted rating equals to 6.342, we represent it on the diagram with a value of 6.3.

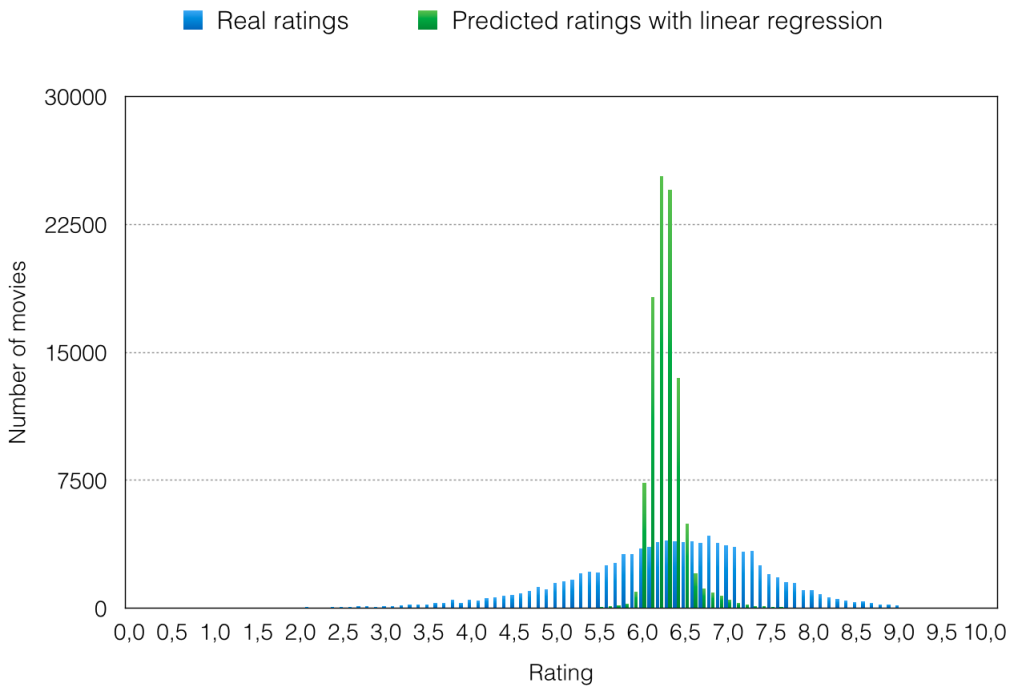

Figure 13: Repartition of the ratings predicted with the linear regression

Even if the linear regression didn't obtain better results, we decided to reduce the number of parameters we had. Indeed, our second goal was to reduce the number of features which describe a movie. With less features, the computation of a regression would be faster and it would allow us to perform other models requiring a long computation time (several days of computation) to a few hours (see Section 3.7). Moreover, we would remove some features which in fact hadn't a real correlation for the prediction of ratings.

In addition to rating predictions, the linear regression implementation provided as output the coefficients of each feature and additional meaningful data. Actually, the software computes a lot of statistical data and performed tests to have a better comprehension of the impact of each parameter.

Thanks to all of these pieces of data, we were able to drop a dozen of features and still get almost similar results. In Section 4, we will explain in-depth in which conditions we removed parameters. The main reason to whether drop a feature or not was the score of the p-value. The p-value is the

probability that a feature is not relevant. Thus, for each feature we looked at the p-value and saw if it was small enough to keep it. This is why we kept only features which had p-value smaller than 0.001. Due to this process, we could use 20 features (without the users ratings feature) instead of 36 features to explain the rating of a movie. We don't taking into account the last feature explicitly because it's the users ratings which is the feature to predict.

Finally, we thought to keep the coefficients we got with the regression and integrate these coefficients with the previous model, the k-NN. We hoped to get better results by applying the coefficients to the features during the computation of the distance between a movie to predict and the training set. This was the last goal we achieved with the linear regression. The last point is explained in details in the next section.

3.5 k-NN Algorithm with Coefficients from Linear Regression

The k-NN we performed gave us promising results but we wanted to increase the accuracy of these results. One of the problem with the simple k-NN we implemented was the fact that every features concerning a movie had the same weight. But we didn't know if the rating of a movie was more influenced by the first actor or the director or something else. This lead us to attempt to give more or less influence to each feature.

As we explained before, thanks to the linear regression, we obtained coefficients for each feature. Actually, these coefficients represent the influence that each feature has on the rating of a movie. By taking the advantage of the coefficients we previously obtained, we were able to put a weight on each feature. Now that we have these coefficients, we had to find a way to apply them during the computation of the k-NN.

As seen previously, we used the Euclidean distance to order the training set depending on the movie we wanted to predict. During this process, the matrix (training set) has been standardised and each feature had the same weight. After the standardisation, all values in the matrix were between 0 and 1. It is this part of the k-NN process we modified. Indeed, we changed a bit the standardisation process to adapt it with the coefficients we got from the linear regression. The formula of the new standardisation becomes

$$X_s = (\frac{X_O - X_{min}}{X_{Max} - X_{min}}) \times w, \tag{15}$$

where X_s is the standardised feature, X_O is the original feature, X_{min} and X_{Max} are respectively the minimum and the maximum value of all the values of that feature and w is the coefficient (weight) of the data's feature from the linear regression.

As you can see, we modified the formula of the standardisation by multiplying the result of a standardisation with the weight of the feature concerned. With this formula, each value of the matrix changed and was no longer between 0 and 1 like with the previous standardisation. Therefore, the value of the distance for each movie also changed. The consequence of that was a change in the order of the nearest neighbors of the movie we had to predict.

The new process is described in Figure 14.

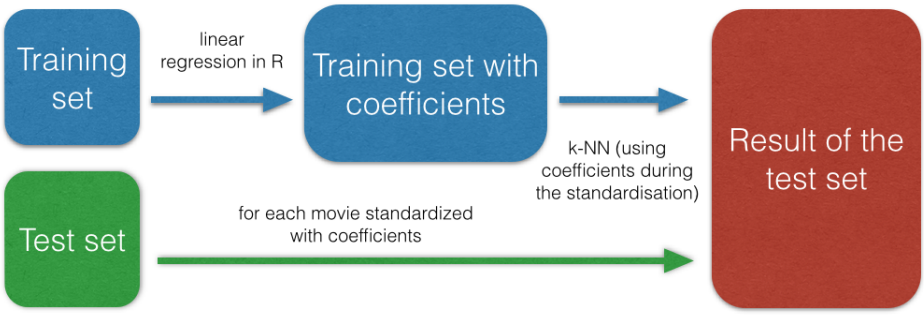

Figure 14: k-NN algorithm with linear coefficients process

After the theory, we adapted our program to take into account the coefficients with the new formula of the standardisation. We performed the computations to get the results and compare it with the previous models. In order to had a better vision of the results, we created a new MSE diagram according to the real ratings shown in Figure 15.

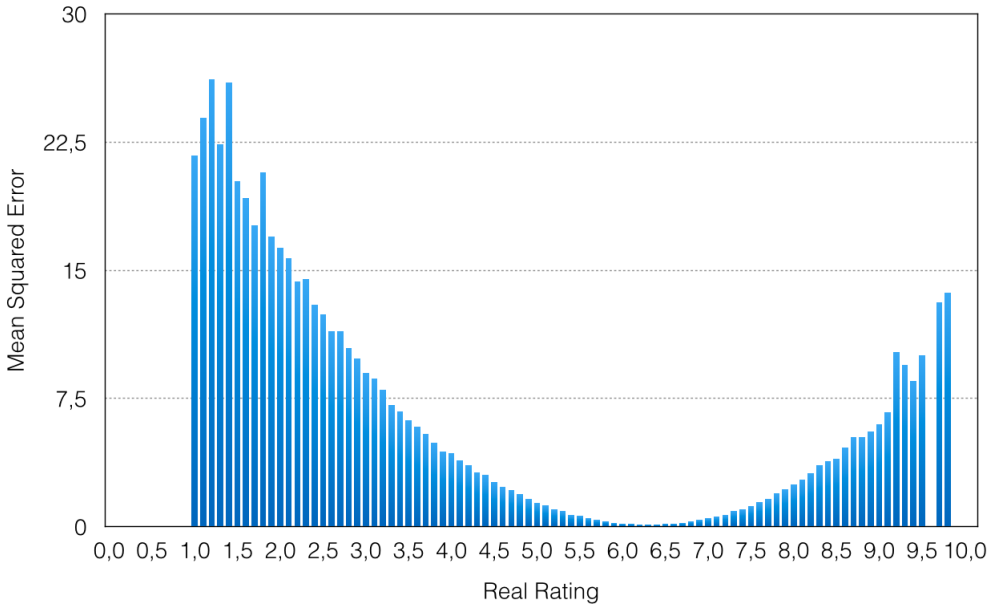

Figure 15: MSE by rating of k-NN algorithm with coefficients from linear regression

As you can see, the bars were more flattened than the previous results. Indeed, the MSE is equals to 1.115 which is lower than the other techniques.

This method opened us a good path to improve the results and pushed us to go further. This gave us the idea that maybe a quadratic regression would fit better with the data than the linear regression. Hopefully, the coefficients of the quadratic regression could be more accurate to have better results.

3.6 Quadratic Regression

After the application of the linear regression, we noticed that the ratings predicted by the linear regression were not well distributed compared to the dispersion diagram of the real ratings (Figure 13). Indeed, estimating the model with a linear function limits the distribution of the ratings predicted. Although the results are better than the prediction with the average of Section 2.2, they are not as good as those obtained with the k-NN with coefficients from linear regression.

Therefore, we applied the quadratic regression to estimate the model with a curve in order to have a distribution of the ratings more similar to the dispersion of the real ratings. Indeed, increasing the degree of a linear function to obtain a polynomial gives us much more parameters (230 coefficients in this case) to evaluate and then, much more accuracy of prediction. The theory of the quadratic regression is explained in Section 2.3 about the polynomial regression. Indeed, the quadratic functions are polynomial functions.

We didn't try to estimate the model with a polynomial of degree higher than 2 because, even if we guess that the results would be better, the goal of this thesis is not to spend time by computing a lot of parameters, it was more interesting to try other models.

As expected, the ratings predicted with the quadratic regression give a better distribution than the ratings predicted with the linear regression. The predicted ratings are less accumulated near the peak and they are more distributed like the real ratings, as shown in Figure 16.

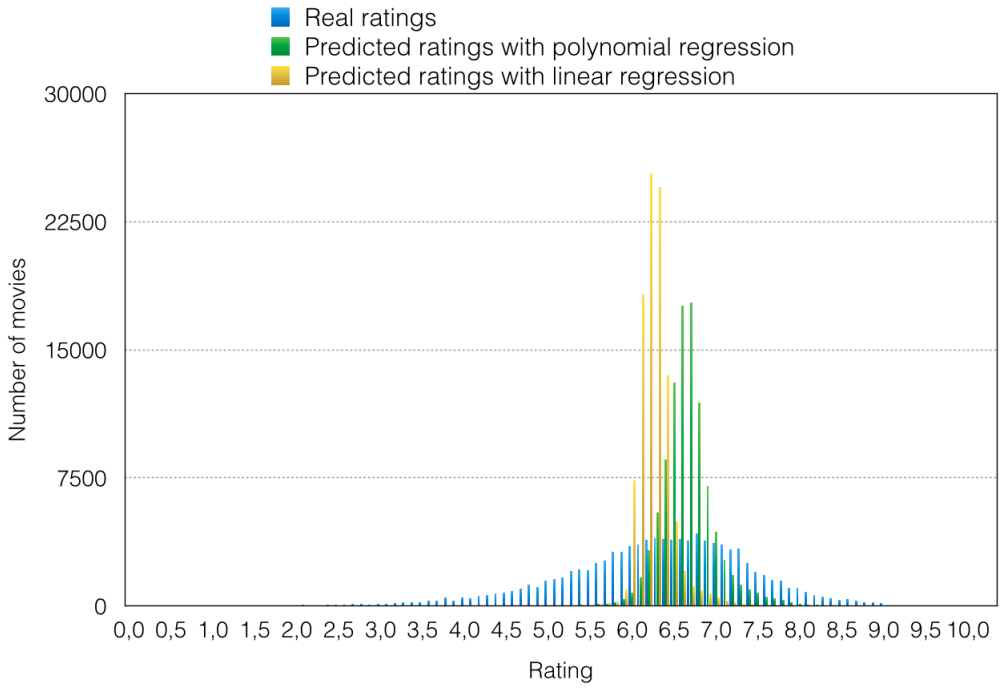

Figure 16: Dispersion diagram of the ratings

Finally, after the experiments, we found out that the predictions given by the quadratic regression are better than the linear regression but not as good as the results given by the k-NN algorithm (with or without the coefficients of the linear regression, as explained in Section 3.5) with a MSE equals to 1.169.

Because of the improvement of the results of the linear regression, we applied these new coefficients in the k-NN algorithm, like with the linear regression coefficients, as explained in the next Section 3.7.

3.7 k-NN Algorithm with Coefficients from Quadratic Regression

This technique is the same than the previous Section 3.5 about the application of the coefficients of the linear regression inside the k-NN algorithm except that, in this case, we use the coefficients provided by the quadratic regression. Therefore, the process and the implementation is quite similar.

However, the computation of the distance of the k-NN algorithm with all the parameters of the quadratic regression takes a long time (several days of computation with 37 explanatory variables). It's why this model has been applied, thanks to the linear regression which allowed us to ignore several useless features to reduce the computation time to few hours, with the 21 remaining features.

When we obtained the coefficients, we applied them to each feature. Quadratic regression gave more coefficients than the number of features as you could expect. Thus we combine features like a quadratic regression would compute. Thanks to this procedure, each coefficient correspond to a specific feature.

As expected, the results provided by this technique improved all the previous results with a MSE equals to 1.099.

3.8 k-Means Clustering

The k-Means clustering would allow us to split the training set into different parts. The idea behind the clustering was to group similar movies together. When we had the training set, we didn't know if all the data belonged to

the same category. Maybe it was wrong to mix movies from the eighties and movies from the fifties because they were totally different and incomparable. Thanks to this intuition the goal was to create distinct groups and then perform the models we have already used before.

To be able to use the clustering model we used the program R. Before trying to implement something we established the steps we had to follow.

First, we needed to know how many clusters we wanted. The answer was that 6 clusters would be a good compromise. We had to be careful because if there were to many clusters, the size to each of them would decrease too much. If it was the case, we would have had some problems to perform a k-NN computation into the clusters. Indeed if the k of the k-NN was greater than the size of a cluster, the method wouldn't get enough movies in the set. The whole specific explanation of why 6 clusters is given in Section 4, we won't detail it more in this section.

Secondly, we had to perform the k-means clustering, as explained in Section 2.4, on the training set. Indeed, our program took two sets in arguments, one was the training set and the second, the test set. At this step, we used the clustering method on the training set to split it into six parts (the size was not specifically equals between them).

When the training set was split, we would integrate each movie in the test set into the right cluster it belongs to. Therefore, we had to use classification techniques to determinate the cluster membership of each movie in the test set.

At the end of this process, we had six clusters containing in each of them a sub training set and a subtest set. As you could understand, this technique allowed us to compare movies of the same category. When we had all the clusters, we could perform the previous models on each cluster to see if we get better results. Therefore, we tried to perform the three k-NN methods of Section 3.3, 3.5 and 3.7 in each cluster

This process is described in Figure 17.

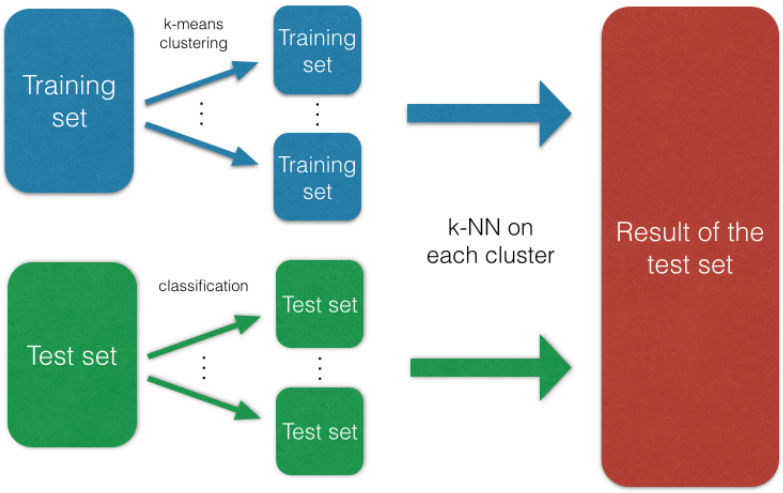

Figure 17: k-NN algorithm with clustering process

One of the advantage in this technique is also the speed. Since we created clusters, the size of the training and the test set in each cluster is smaller than the original ones. The consequence is a faster computation because there are less movies. Thus, the program has less distances to calculate to predict the rating of a movie.

To conclude this section, we could say that create clusters provides better results than just performing a k-NN without creating clusters before. All the detailed results are presented in Section 4.

3.9 Distance-Weighted k-NN Algorithm

The theory of the k-NN algorithm, presented in Section 2.1, taught us there are two ways to calculate the average of the ratings of the k-nearest neighbors. The first way has been applied during all the process explained in this section, it consists to simply calculate the arithmetic average of the rating neighbors. The second way is to weight the influence of the neighbors according to their distance, the nearest as the most influent. We tried this method after all the experiments already performed with the simple average and we found out that the weighted average gave us much better results.

At first, we didn't think about modify the influence of the neighbors and we didn't expect an increase of performance that much. Therefore, we started

again all the experiments with this little modification of the algorithm: apply a weight of $\frac{1}{d}$ to each k-NN with d equals to the distance value. Thanks to this weight, the nearest neighbors have more influence than the farthest ones.

Moreover, influencing the impact of the neighbors according to their distance gives less importance to the value of k which has a strong impact on the result. This value is only based on the training set, it's why we predict the optimal k by cross validation and we hope this value will be optimal for the test set as well. Therefore, the decreasing weights given to the neighbors limit the impact caused by a non optimal value of k.

All the results obtained by this method are presented in Section 4.

4 Result

In this section, we detail all the results we obtained and we deepen some choices we have made. Then, we analyse the results of the experiments to determine if these results are significant. Furthermore, we explain our implementations of the methods we saw in Section 3 and how we retrieve the data from the IMDb website. Finally, we take a future movie as an example to predict the rating with our best technique.

First of all, we begin by explaining the way we performed the tests and retrieved all the results we needed. We decompose it into two phases; the test phase and the result phase.

4.1 Test Phase

After describing and trying each technique to verify if we were in a right direction, we decided to create a complete batch tests and analyse it afterwards. As we had 15 different techniques, we had to be carefully of the time it could take.

In this 15 techniques, we had the baseline technique, the two functions from the regression models, the 6 k-NN techniques and the 6 clusterings techniques. Figure 18 is a good representation of all the tests we have done.

In Figure 18, all the leafs of the tree are techniques we described previously, implemented and tested. The schema is to be read from top to bottom. Thus, the root of the tree, the top, is the dataset which contains 60,089 movies with all the features to describe them. As you can see, the dataset is split randomly into two subsets, the training set (90%) and the test set (10%). Then, we used these subsets as input for every nodes. So, following the arrows, you can observe that we tested all the techniques already explained before.

Moreover, as shown on the schema (top right), we performed these tests 17 times. This means that each instance of the program contains another training and test set. So even if the whole dataset is always the same, the two subsets are different because they are randomly generated at each instance.

For each technique we used, we obtained the mean square error of the rating predictions. Actually, we compute the MSE of each rating movie predicted. Since we didn't perform one time each technique but 17 times, we

had to merge the results of the same technique together.

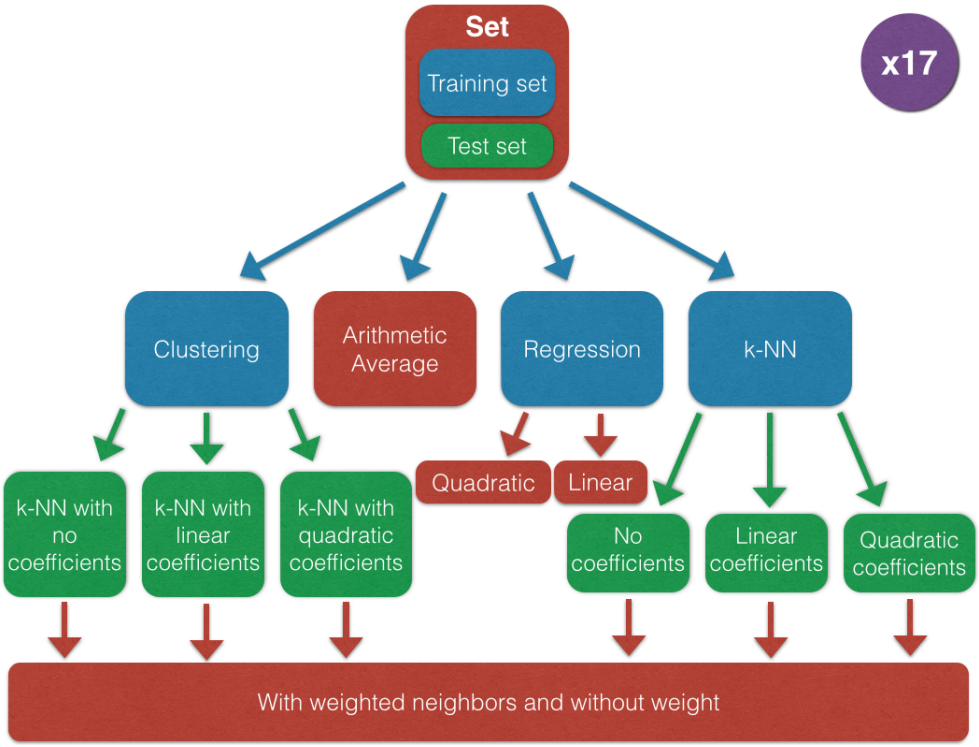

Figure 18: Summary of the all techniques

4.2 Result Phase

During this phase, we merge all results from each technique together. The result of it is represented in Figure 19. You can see that we kept the real rating of the test set to be able to evaluate the error of the prediction. When we predicted all the rating movies, we compute the MSE which is the average of the square errors of movies.

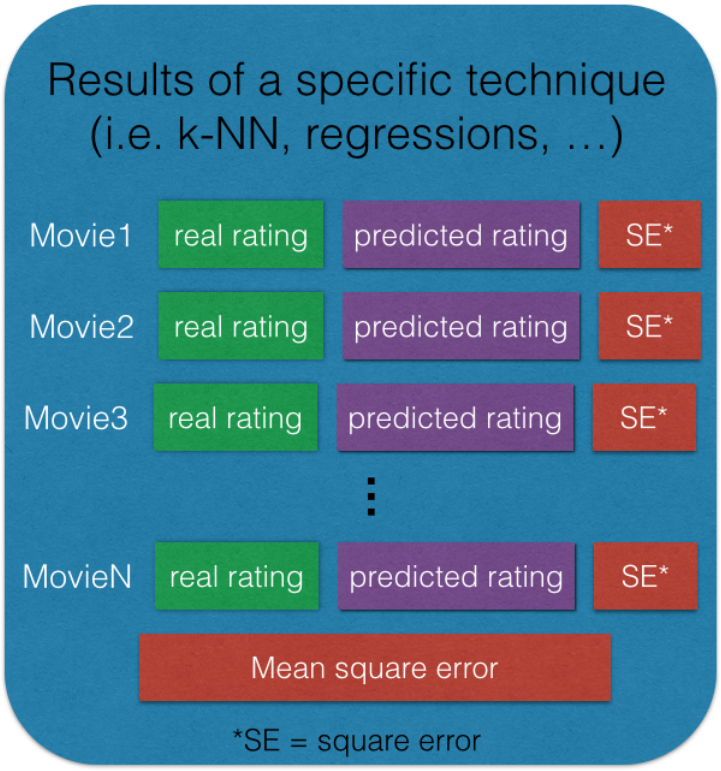

Figure 19: A result file

The final output of the results phase were 15 files containing in each the results of each techniques. Each file predicts the ratings of 102,136 movies. Indeed, the size of each test set was 6,008 movies and we computed 17 different tests ($102,136 = 17 \times 6,008$).

By generating 17 random subsets (training set + test set) from the same set, we were able to increase the number of predicted ratings. This allowed us to have a more reliable MSE to be able to compare each technique with other ones.

Thanks to these two phases, we built a spreadsheet to have a global view of the MSE of each result. You can see the spreadsheet in Figure 20. As you saw, the 15 techniques are represented. We decided to split the table in two columns. One column with the techniques using weighted neighbors and another without weights. You can observe that the weighted neighbors approach seems to give better results than the other one.
All the techniques have been explained in Section 3 and some of them will

be detailed in the next subsection.

The lowest MSE for these tests is 0.9978 with the clustering with the distance-weighted k-NN algorithm with coefficients from quadratic regression in each cluster. The higher MSE is the one with the baseline model (1.25).

Techniques	**No weight**	**Distance-Weighted**
Arithmetic average	1.2500	/
Linear regression	1.1976	/
Quadratic regression	1.1690	/
k-NN	1.1295	1.0371
k-NN with coefficients from linear regression	1.1150	1.0204
k-NN with coefficients from quadratic regression	1.0990	1.0047
Clustering with a k-NN in each cluster	1.1037	1.0138
Clustering with a k-NN with coeff. from lin. regr. in each cluster	1.1036	1.0125
Clustering with a k-NN with coeff. from quadr. regr. in each cluster	1.0895	0.9978

Figure 20: MSE of each technique

We concluded that the lower MSE is 20% lower than the higher MSE. But we had to be very careful with this kind of hasty conclusion because we have to perform a statistical hypothesis test to determinate if a technique is significantly better than another one. This test is the subject of the next subsection.

4.3 Statistical Hypothesis Testing

After the explanations of the experiments and the presentation of the results obtained with all the methods, we will present a statistical hypothesis test applied to the results of two of our methods. The goal of the test is to determinate if the results obtained by these two methods are significantly different and, therefore, that one is better than the other one to predict the movie ratings based on a dataset from IMDb.

Indeed, even if the results give us an evaluation of the performance of the methods, we can't say that a method is better than another without a statistical hypothesis test. In this case, we only perform one test in order to know if a good model like the quadratic regression (Section 3.6) gives significantly better results than the benchmark computed with the arithmetic average (Section 3.2).

This test is based on the results obtained with one experiment where we applied the two methods on the same training set and test set. The statistical hypothesis test evaluates if the MSE's obtained with the two methods are similar or significantly different. Since the two methods are performed on the same test set, we are in the case of a comparison of the means of two sets of paired samples. Therefore, we need to apply a Student's t-test [7] for paired samples.

In this case, the theoretical deviation of the MSE's is unknown, therefore, we have to estimate it with an experimental deviation with the formula

$$S_D^2 = \frac{1}{n-1} \sum_{i=1}^{n} (D_i - \overline{D})^2, \tag{16}$$

where S_D^2 is the experimental deviation, n is the number of movies in the test set, D_i is the difference of the squared errors of the test i between the two methods, $\overline{D}$ is the mean difference of the squared errors between the two

methods of all the movies in the test set.

Now that we have the experimental deviation, we can compute the t-test value according the following formula (17), in our case the value of μ_D is equals to zero because we want to know if the difference of the MSE of the two methods is equals to zero.

$$t = \frac{\overline{D} - \mu_D}{\sqrt{\frac{S_D^2}{n}}}, \tag{17}$$

Thanks to these formulas, we can start the statistical hypothesis test and set the hypothesis zero as the hypothesis where the MSE of the prediction with the arithmetic average is equal to the MSE with the quadratic regression,

$$H_0 : \mu_D = MSE_1 - MSE_2 = 0, \tag{18}$$

where MSE_1 and MSE_2 are respectively the MSE obtained with the prediction with the arithmetic average and the MSE obtained with the quadratic regression.

The hypothesis H_0 is rejected if our t value is equals to or bigger than the Student value $t_{1-\alpha/2,n-1}$ where α is the signification level of the test,

$$|t| \geq t_{1-\alpha/2,n-1}. \tag{19}$$

With the results obtained with the two methods and the formulas presented above, we obtained the following value:

$$\overline{D} = 0.054 \quad S_D^2 = 2.7429 \quad t = 2.5315 \tag{20}$$

with $n = 6008$.

We chose a signification level of $\alpha = 0.05$, therefore, the Student value $t_{1-\alpha/2,n-1}$ is equal to 1.96.

Finally, according to the equation (19), we can reject H_0 and say that the MSE of the two methods are significantly different. Therefore, since the MSE obtained with the quadratic regression is lower than the MSE obtained with the prediction with the arithmetic average, we proved that the quadratic regression is a significantly better technique to predict the movie ratings with

a dataset from IMDb.

4.4 Algorithm Tuning

This section explains choices we had to make in order to solve some problems. First, we explain how we dropped some features with the linear regression and then why we chose to split the dataset into 6 clusters.

4.4.1 Drop Features

Thanks to the linear regression, we were able to drop some features without having significant impact on results and obtain lower MSE by using coefficients. In Section 3.4, we explained that we removed some features mostly because of the computation time. When we used the `lm` command on the R program to obtain the coefficients from the linear regression, we obtained lots of relevant data.

As you can see in Figure 21, a part of the summary of the results of the `lm` method is shown. It represents a table containing 6 columns. The first column is obviously all the features describing a movie, the explanatory variables. Also, there is the intercept which doesn't belong to any feature. This intercept was only useful when we computed the result with the formula of the linear and quadratic regressions.

The second column contains the estimation of the coefficient of each feature. These values will be used for the formula of the regressions and k-NN with coefficients. The third column is the standard error of the coefficient estimate. The standard error is a relative to the value of the coefficient. If the value is low, it means that the error is small which is good.

The fourth and fifth column are correlated. The t-value of the coefficient estimate is used to calculate the fifth column, the p-value. This column is very important for the decision process of dropping a feature. The p-value is the probability that the coefficient is not relevant. Thus, the lower the p-value, the more meaningful the coefficient.

Finally, the last column is represented by stars. The stars gave a good representation of the significant levels of the coefficients. So, 3 stars means that the p-value is between 0 and 0.001 and so on. You can look at the

bottom of the figure for the legend.

We chose to remove every features which didn't have at least 3 stars. It means that we dropped every coefficients which had a p-value higher than 0.001. Thanks to this process, we were able to drop dozen features by keeping only reliable and meaningful features.

```
Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         1.515e+01  4.168e-01  36.355  < 2e-16 ***
awardMovie          3.960e-02  1.341e-03  29.537  < 2e-16 ***
numRatingUser       4.416e-06  1.110e-06   3.979 6.93e-05 ***
numReview          -2.401e-05  3.353e-04  -0.072 0.942920
numCritic           1.535e-03  4.886e-04   3.142 0.001677 **
numMeta            -1.479e-02  2.089e-03  -7.077 1.48e-12 ***
year               -3.773e-03  2.918e-04 -12.932  < 2e-16 ***
act1NumMovie       -1.044e-04  7.407e-05  -1.410 0.158532
act1NumProd        -2.066e-04  5.511e-04  -0.375 0.707724
act1NumDir          7.266e-04  3.820e-04   1.902 0.057150 .
act1NumAward        4.240e-03  8.333e-04   5.088 3.64e-07 ***
act1Birth          -3.248e-05  1.201e-04  -0.270 0.786878
act1NumWriter      -1.506e-05  5.575e-04  -0.027 0.978443
act1NumNomination  -3.153e-03  8.845e-04  -3.565 0.000365 ***
act2NumMovie        1.512e-04  6.874e-05   2.200 0.027808 *
act2NumProd         7.846e-04  6.280e-04   1.249 0.211512
act2NumDir          1.449e-04  4.162e-04   0.348 0.727643
act2NumAward        3.420e-03  1.043e-03   3.279 0.001041 **
act2Birth          -4.475e-04  1.106e-04  -4.048 5.18e-05 ***
act2NumWriter      -7.241e-04  6.766e-04  -1.070 0.284560
act2NumNomination  -1.923e-03  1.048e-03  -1.835 0.066518 .
act3NumMovie       -2.103e-04  6.365e-05  -3.305 0.000951 ***
act3NumProd         1.702e-04  6.962e-04   0.245 0.806829
act3NumDir         -2.155e-04  4.091e-04  -0.527 0.598358
act3NumAward        2.863e-04  1.324e-03   0.216 0.828782
act3Birth          -2.500e-04  1.294e-04  -1.932 0.053336 .
act3NumWriter       4.021e-04  8.488e-04   0.474 0.635690
act3NumNomination  -4.918e-04  1.318e-03  -0.373 0.708968
dirNumMovie        -4.717e-04  1.256e-04  -3.755 0.000174 ***
dirNumProd         -3.969e-04  1.464e-04  -2.711 0.006701 **
dirNumDir           3.593e-04  9.399e-05   3.823 0.000132 ***
dirNumAward         4.431e-03  1.038e-03   4.269 1.97e-05 ***
dirBirth            6.941e-04  1.329e-04   5.225 1.75e-07 ***
dirNumWriter       -1.946e-03  1.641e-04 -11.855  < 2e-16 ***
dirNumNomination    1.010e-02  1.199e-03   8.428  < 2e-16 ***
writerNumMovie     -4.447e-04  1.493e-04  -2.979 0.002892 **
writerNumProd      -7.693e-04  2.312e-04  -3.328 0.000876 ***
writerNumDir        2.842e-06  1.380e-04   0.021 0.983570
writerNumAward      1.451e-03  1.266e-03   1.146 0.251702
writerBirth        -7.523e-04  8.999e-05  -8.359  < 2e-16 ***
writerNumWriter     6.298e-04  8.859e-05   7.109 1.18e-12 ***
writerNumNomination 2.282e-03  1.416e-03   1.611 0.107233
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 21: R output for linear regression

4.4.2 Number of Clusters

During the clustering process, we had to give a number of clusters as input. The number of clusters we generate will directly influence their size and then the results. Indeed, the more clusters, the smaller the size of each cluster.

The number of clusters could have an impact on the results because it could be possible that the size of each cluster which we performed techniques on, such as k-NN, is too small for the prediction. Actually, if the size of the cluster is too small, the cross validation into this cluster couldn't be optimised and the k-NN couldn't be efficient; the process decision to select the k neighbors from the training set will often be the same. So, the prediction will be often the same between all the tests.

For this reason, we had to make another design choice and had to choose a number of clusters. To achieve this task, we used a script which call some useful functions in R such as `kmeans` (compute clustering), `plot` (generate graphs) and `clusplot` (generate a graph to represent the result of the clustering).

Our decision process has been made through one specific graph we created, Figure 22. The horizontal axis corresponds to the number of clusters and the vertical axis help us to decide the number of clusters to choose. The sum of squared errors (SSE) is defined as the sum of the squared distances between each member of a cluster and its cluster centroid. SSE can be seen like the measure of the error.

We can observe that the within groups SSE which is the vertical axis decreases when the number of clusters increases. As you can see, the curve follows a decreasing function. After 6 clusters, the curve seems to be a linear slope, meaning that it won't be a huge advantage to take more than 6 clusters.

Thanks to this graph, we fixed our clusters number to 6. It seemed a good trade-off between the error deviation and the number of clusters.

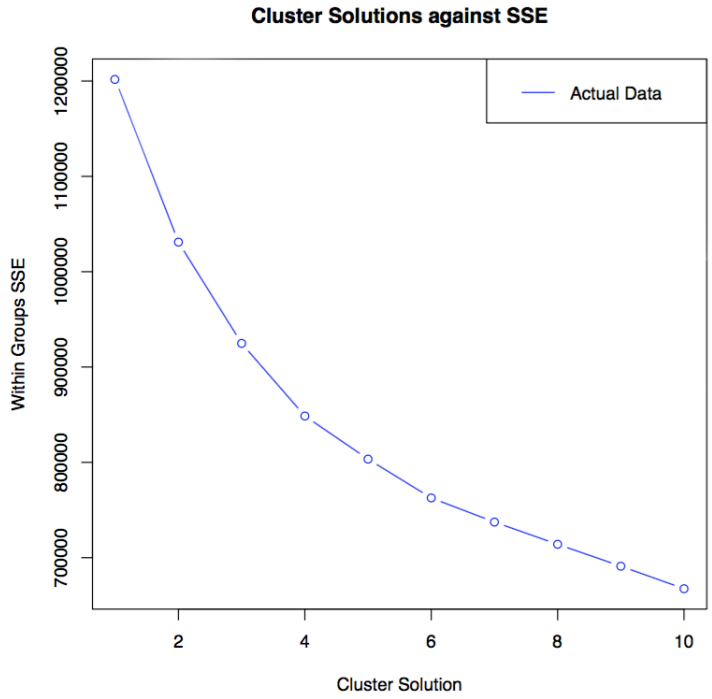

Figure 22: Clusters Solution Against Sum of Squared Error

We finished by generating a graph displaying each movie of the dataset on a scatter plot, Figure 23. We used the principal component analysis (PAC) [4] to put each movie on the plot. PAC is a powerful statistical instrument. The theory of the PAC is out of the scope of this thesis. Nevertheless, in our case, the PAC is used to convert every movie described by several components (features) into two principal components. Since we explained as well as possible a movie with two components, we can represented a movie on a 2D-graph.

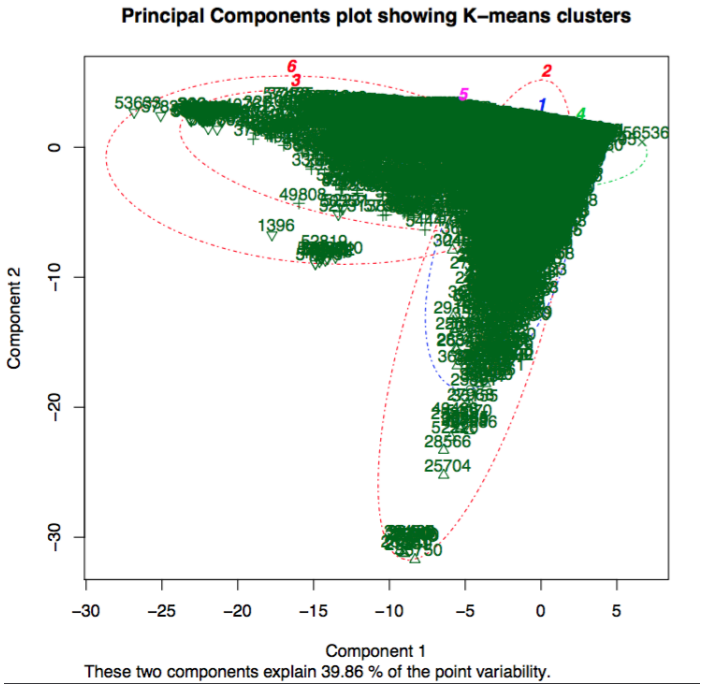

Figure 23: Principal Component Analysis on the Data Set

4.5 Best technique

This section contains detailed diagrams about our best technique to predict the movie ratings; the distance-weighted k-NN algorithm with coefficients from quadratic regression using clustering. Indeed, this technique obtained the lowest MSE compared to the other ones as presented in Figure 20.

Figure 24 shows the mean squared error according to the real rating. In this diagram, each column represents the value of a MSE. It is computed between all the instances having the same real rating. We notice that the value of the MSE's are lower than those on the other diagrams shown in Section 3.

Then, the dispersion diagram of the ratings predicted by our best technique is shown in Figure 25. We can see that the distribution of the ratings predicted by our best technique is more similar to the distribution of the real ratings than the other techniques.

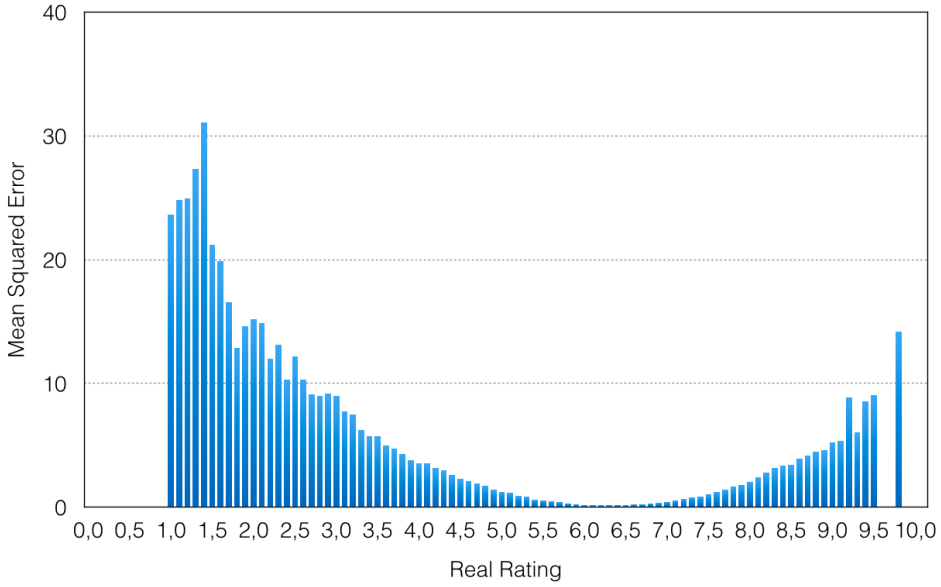

Figure 24: MSE by rating with clustering with our best technique

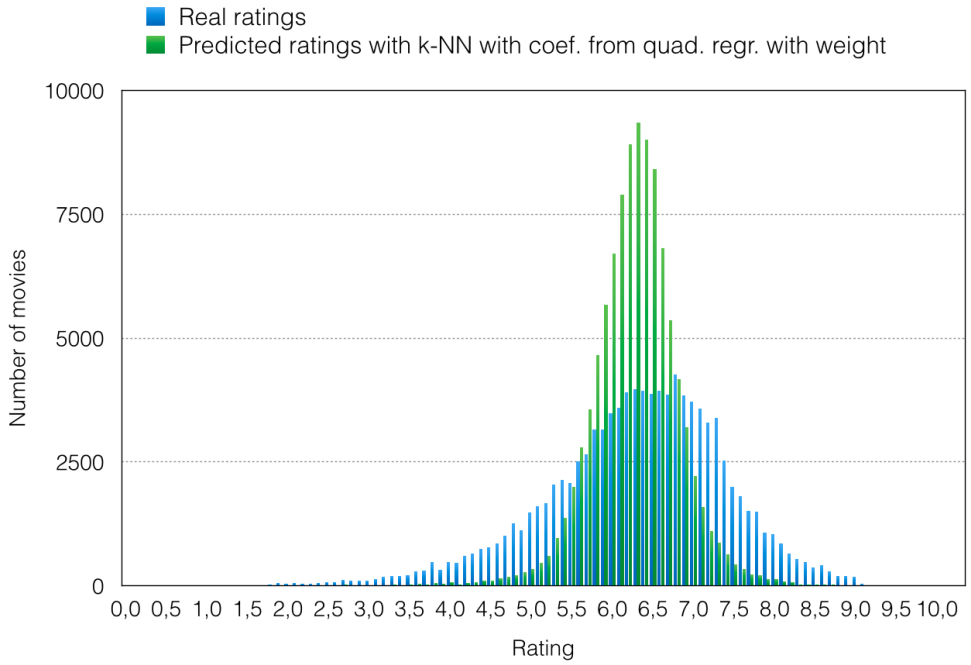

Figure 25: Dispersion diagram of ratings predicted with our best technique

4.6 Implementations

This subsection explains the details of our implementations of the techniques explained in Section 3. All the programs developed in the scope of this thesis have been developed in Python and are available on the link: github.com/nrvthesis.

The first program developed was the program to parse the website of Internet Movie Database in order to retrieve all the data about movies and to store them in a text file. The program was in charge of downloading each HTML page about movies and to parse them to fill the text file. The computation time of this program was about 1 week to retrieve the data of 60,089 movies.

The second program we developed was the program implementing the k-NN algorithm. Due to the simplicity of the algorithm, the first version of the program has been achieved in few hours and we obtained the first results. The program implemented the Euclidean distance function (Equation 1), the standardisation function (Equation 3) and the average of the k-nearest neighbors. Later, with the techniques using the k-NN algorithm with the coefficients of the linear or quadratic regression, the computation time of the program started to become huge (more than 1 day with the k-NN with coefficients from the quadratic regression). Therefore and because of the growing complexity of the techniques, we decided to re-implement the program from scratch to handle the multi-threading, the cross-validation and the management of the coefficients in the computation of the distances.

For the linear and the quadratic regression, we used a program called R to provide the coefficients and other relevant data about our model. We automatised the computation of the coefficients for each test and we computed the rating predicted with the function provided by R with another program.

Finally, for the clustering, we implemented the algorithm to split the movies into 6 different clusters and perform other techniques into each of them. We created another R script to automatised the k-means clustering and the classification process thanks to the respectively functions `kmeans` and `cl_predict`.

Thanks to these programs, we developed a final program to launch all the other programs in such a way that the programs work together to perform the techniques explained in Section 3.

4.7 Prediction of a future movie

To conclude the result section, we decided to predict the rating of the movie "Star Wars: Episode VII - The Force Awakens" to be released in December 2015 with our best method to predict a movie which has not been released yet.

First, we retrieved all the relevant features of the movie on IMDb to perform our technique, the features are shown in Figure 26. Then, we applied the distance-weighted k-NN algorithm with the application of the coefficients of the quadratic regression using the k-means clustering to obtain the predicted rating.

#Movie Director	7
#Production Director	43
#Direction Director	13
#Writing Director	21
#Award Director	15
#Nomination Director	29
Date of Birth Director	1966
#Movie Writer	7
Year of Prod	2015
#Production Writer	43
#Writing Writer	21
#Award Writer	15
Date of Birth Writer	1966
Date of Birth Actor 2	1951
#Nomination Actor 1	31
#Award Actor 1	0
#Award Actor 2	10
#Direction Actor 1	0
#Movie Actor 2	269
#Movie Actor 3	87

Figure 26: Statistics about "Star Wars: Episode VII - The Force Awakens"

The meaning of the different features present in Figure 26 are the same as the features explained in Figure 7 in Section 3.1.

Finally, the rating predicted by our best technique for "Star Wars: Episode VII" is 6.8/10, this rating has been rounded to fit the format of the rating of IMDb.

5 Conclusion

The goal of this thesis was to forecast the success of movies based on IMDb. To achieve this goal, we applied several machine learning algorithms to predict the movie ratings. The main techniques used were the k-NN algorithm, the linear regression, the quadratic regression and the k-mean clustering. We performed a lot of experiments with several implementations of these methods and we combined them to obtain the best prediction of the ratings.

The first step of the thesis was to collect the data from the website IMDb in order to build a complete dataset. After one week of computation, we obtained a database containing data about more than 60,000 movies thanks to a program parsing the website. The dataset containing almost all the movies available on IMDb, we could start using the different machine learning methods to predict the movie ratings.

To apply the different techniques and evaluate their performances, we had to split the whole dataset into a training set containing 90% of the dataset and a test set containing the remaining 10%. Indeed, the machine learning algorithms used in this thesis use the training set to predict the rating of the movies from the test set. Then, the predicted ratings of the test set are compared to the real ratings observed in order to obtain the errors on the predictions and to compute the mean square error of the method which is used to evaluate the performance of a technique.

The first prediction technique used was to predict the rating of each movie from the test set with the arithmetic average of all the real ratings from the training set. This method is very simple and we didn't expect good results from it but the main goal of this technique was more to obtain a performance benchmark allowing to compare the performance of the other techniques than to predict the movie ratings with a good accuracy.

The second method applied was the k-NN algorithm, a popular lazy learning algorithm used in machine learning. This method uses the Euclidean distance function to find the movies inside the training set that are the more similar to the movie to predict. Then, the algorithm computes the average of the rating of the k most similar movies called the k nearest neighbors. Finally, this mean is assigned to the movie to predict as the predicted rating. We used the cross-validation technique to find the optimal value of k, the number of neighbors to take into account for the prediction. The influence of the neighbors can be modified according to their distance of the movie to

predict by applying a weight inversely proportional to the distance such that the movie with the lowest distance has the most influent rating in the computation of the average. For all the techniques using the k-NN algorithm, we performed the computation of the average of the k nearest neighbors with and without the distance-weighted neighbors to evaluate the difference of performance. As expected, the results obtained with the k-NN algorithm were much better than the results obtained with the first technique providing the benchmark. What's more, the distance-weighted k-NN algorithm gave us even better results than the version without the weights.

The next technique used was the linear regression with which we estimated the model with a linear function. First, the linear regression gave us the opportunity to removed some features about movies contained in the dataset deemed irrelevant for the prediction of the rating. Therefore, we reduced the number of features about movies used to predict the rating. At the beginning we had 37 different features about movies, including the rating, and we reduced this number to 21 by removing those with a big p-value which is the value evaluating if a feature is relevant or not. Thus, we performed all the experiments with these 21 features. Then, the linear regression provided us the coefficient of each feature to predict the movie ratings thanks to the linear function. The performance of this function to predict the rating was not as good as the prediction of the k-NN algorithm but better than the prediction with the arithmetic average. Moreover, we noticed that the dispersion diagram of the ratings predicted with the linear regression didn't have a good distribution like the dispersion diagram of the real ratings.

For the fourth technique, we applied the k-NN algorithm with the coefficients obtained thanks to the linear regression in order to influence the computation of the distance of the k-NN algorithm. The application of these coefficients allowed us to give more importance to the relevant features and to obtain the nearest neighbors thanks to a distance value more significative. This method using the two previous techniques gave us a MSE lower than all the previous techniques meaning that it is the most efficient technique.

Because of the good results obtained with the linear regression and the usefulness of its coefficients, we chose to apply the quadratic regression as fifth model to estimate the model with a quadratic function. This regression gave us much more coefficients than the linear regression to predict more precisely the movie ratings. The application of the quadratic function gave better results than the linear function but the results of the k-NN algorithm with or without the coefficients of the linear regression are still better than

the ratings predicted by the quadratic regression.

The sixth technique applied combined the k-NN algorithm with the coefficients of the quadratic regression. Like the fourth technique, the application of these coefficients on the features of the movies provides a better evaluation of the distance between the movie to predict and the movies inside the training set. This technique was the longest in duration of computation because the k-NN algorithm computes a lot of distances during the all the process and the computation of the distance becomes a long operation because of the large number of coefficients provided by the quadratic regression. However, this technique gave better results than all the previous techniques.

The last techniques applied in this thesis use the k-mean clustering method. The k-mean clustering is not a regression method to predict a value like all the previous techniques, its goal is to split the dataset into k clusters and group the similar movies together in the same clusters. This method allowed us to perform the k-NN algorithm with and without coefficients with a smaller training set corresponding to the cluster whose the movie to predict looks very much like other movies in the cluster. Therefore, the computation time of the techniques applied with the clustering is reduced which is useful for a technique like the k-NN algorithm with the coefficients of the quadratic regression. Moreover, all the techniques applied with the clustering gives better results than the same techniques applied without the clustering.

The results are computed according 17 experiments with the same dataset. For each experiment, the dataset is randomly split into a training set and a test set. According to these experiments, the best technique to predict the movie ratings is the distance-weighted k-NN algorithm with the application of the coefficients of the quadratic regression using the k-means clustering.

The results of the application of all the techniques presented are illustrated with diagrams. Moreover, a statistical hypothesis test has been applied between the average technique and the quadratic regression technique to demonstrate that the difference of the results obtained by these two methods is significative and then, in our case, that the prediction with the quadratic regression is significantly better than the prediction with the average. Finally, we predicted the rating of the movie "Star Wars: Episode VII - The Force Awakens" to be released in December 2015 with our best method to predict a movie which has not been released yet.

Future work would apply a polynomial regression with a degree higher

than 2. Indeed, as we noticed with the linear and the quadratic regression, raising the degree of a polynomial improves the accuracy of the rating predicted and provides more coefficients which can be used with the k-NN algorithm to lead to even better results. Another way would be to increase the amount of data present in the dataset in order to apply the techniques with larger training sets which would probably improve the performances of all the techniques.

References

[1] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[2] T. M. Mitchell, *Machine learning*. McGraw-Hill, 1997.

[3] I. H. Witten and E. Frank, *Data mining : practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, 2000.

[4] A. Berson and S. J. Smith, *Data warehousing, data mining, and OLAP*. McGraw-Hill, 1997.

[5] [Online]. Available: https://apandre.files.wordpress.com/2011/08/kmeansclustering.jpg

[6] A. Cornuejols and L. Miclet, *Apprentissage artificiel : concepts et algorithmes*. Eyrolles, 2002.

[7] I. Gijbels, *Probabilités et Statistiques 2*. Diffusion Universitaire Ciaco, 2004.

[8] R. S. Michalski and J. G. Carbonell, *Machine Learning : An Artificial Intelligence Approach*. Morgan Kaufmann, 1986, vol. 2.

[9] M. L. Brodie and J. Mylopoulos, Eds., *On knowledge base management systems : integrating artificial intelligence and database technologies*. Springer, 1986.

[10] J. S. Kowalik, *Knowledge Based Problem Solving*, J. S. Kowalik, Ed. Prentice Hall, 1986.

[11] P. H. Winston, *Artificial intelligence*. Addison-Wesley, 1992.

[12] M. Marvin, *The society of Mind*. Heinemann, 1985.

[13] R. S. Michalski, Ed., *Machine learning and data mining : methods and applications*. Wiley, 1999.

[14] N. J. Nilsson, *Artificial intelligence : a new synthesis*. Morgan Kaufmann, 1998.

[15] E. Charniak, C. K. Riesbeck, and D. Mcdermott, *Artificial Intelligence Programming*. Lawrence Erlbaum, 1980.

[16] B. Duboulay and L. Steels, *Advances in artificial intelligence.* Elsevier, S.l., 1987.

[17] J.-P. Delahaye, *Formal Methods In Artificial Intelligence.* North Oxford academic S.l, 1987.

[18] R. Hawley, Ed., *Artificial intelligence programming environments.* Ellis Horwood, 1987.

[19] H. Farreny and M. Ghallab, *Elements D'Intelligence Artificielle.* Hermès S.l., 1987.

[20] D. Partridge, *Artificial Intelligence : Applications In The Future of Software Engineering.* Ellis Horwood, 1986.

[21] M. R. Genesereth and N. J. Nilson, *Logical foundations of artificial intelligence.* Morgan Kaufmann, 1987.

[22] R. Forsyth and R. Rada, *Machine Learning : Applications In Expert Systems and Information Retrieval.* Ellis Horwood, 1986.

[23] S. Minton, Ed., *Machine learning methods for planning.* Morgan Kaufmann, 1993.

[24] B. S. Everitt, S. Landau, and M. Leese, *Cluster analysis.* Edward Arnold, 2001.