PREDICTING IMDB SCORES

Using IMDb 5000 Dataset

Final Project

MA429: Algorithmic Techniques for Data Mining
Candidates: 84887, 86466, 89765

Executive Summary

IMDb scores can be seen as a measure of movie quality. A regression as well as a classification setting were applied on a dataset comprising data on approximately 5000 movies.
Linear regression was implemented to draw inference between the independent and dependent variables followed by an array of classification methods which predicted the IMDb score within a range. The range of scores were called bins.

The technique which stood out was “Boosting”.
It provided an accuracy of 82.33%. Since the classes were highly imbalanced Kappa was also used as a measure of performance. Its value was 62.28% which indicates the model was performing fairly well across all the different binned scores.

A few other noticeable techniques include:

1. Multinomial Logistic Regression
2. SVM – Linear
3. SVM – Polynomial
4. SVM – Radial
5. Linear Discriminant Analysis
6. Random Forests
7. K-Nearest Neighbours

Table of Contents

1 Introduction

1.1 Motivation

Before a movie is published, there is one crucial question: Will it be a good movie? Many people are affected by the outcome. The customers in the cinema want to see a good movie, the actors want to push their career with a good movie, and the production company as well as the film crew wants to produce a good movie to make profits or, at least, break even. One way to assess the quality of a movie is the score on the Internet Movie Database (IMDb) where everyone can contribute via voting[1]. The score ranges from 1 (worst) to 10 (best). Although there is some criticism of the IMDb scores (for example a bias towards male voters[2]), they generally seem to reflect the average opinion of people (given there are enough votes for a movie). With certain limitations, this was also confirmed by Baugher and Ramos (2016).

The prediction of IMDb scores has already been subject of previous research. Oghina et al. (2012) used social media data from YouTube and Twitter to analyse IMDb score prediction. They concluded that their regression approach yielded a high forecasting performance. Other works have used predictors directly provided by IMDb (such as actors of a movie or its genre): While Hsu et al. (2014) concluded that the models they applied were able to predict IMDb scores well, an earlier study of Saraee et al. (2004) suggested that score prediction using IMDb data only is rather difficult.

A dataset which combines social media data (from Facebook) and IMDb data (including scores) is the IMDB 5000 dataset consisting of 5043 movies. It was analysed as part of a competition on Kaggle[3]. The authors converted the IMDb scores into four score ranges reflecting different quality levels and predicted the range to which a movie belongs. They used three methods for their classification problem: Trees, k-Nearest Neighbors and Random Forests. Their approach resulted in an accuracy of about 77% for predicting the correct quality level of movies.

1.2 Goal and Limitations

This work analyses the previously mentioned IMDB 5000 dataset. The project aims to improve the insights of IMDb score prediction. To achieve this, two approaches were incorporated. An extended classification setting was established to forecast the quality of a movie where the results of the authors of the Kaggle competition were used as a benchmark. Pure predicting is not able to explain the link between predicting variables and the dependent variable in detail. Yet, this might be of interest for someone who has to decide on different aspects of a movie before it is produced (such as the budget for a movie or the principal actors). Therefore, an inference setting was applied as well to gain further insights through regression analysis.

The IMDB 5000 dataset contains many missing values. Therefore, the cleaning of the data demanded a lot of attention. It was often not possible to gather those missing values. About 25% of the observations in the dataset have missing values for some variable. Since the

[1] IMDb (2018a)
[2] Wired (2018)
[3] Kaggle (2018)

dataset only comprises 5043 observations, those incomplete observations could not simply be dropped. Otherwise, the dataset would have been very small, and a lot of information would have been lost. That is why several assumptions for missing value imputation had to be made. These are specified in chapter 2. Moreover, it was found that some values in the dataset might be incorrect. For example, the dataset specifies the director of a movie and the likes on his/her Facebook page. The director James Cameron is listed and has zero likes according to the dataset. Checking his Facebook page reveals that this is not true. In addition, there is a number incorrect values for two other variables, box-office gross and budget of a movie. Both variables are stated in US dollar. Yet, for some of the movies in the dataset these values are stated in local currencies. These values would have to be converted to US dollars. An exploratory analysis of the two variables indicated that only a few movies seem to be affected by this. Since it was not possible to check all values in the dataset on their correctness, the existent data was used as is knowing that it might be flawed.

1.3 Structure of Report

In the next chapter, the dataset and the pre-processing approach are described. The performance measures for both, the regression as well as the classification setting, are presented in chapter 3. Then, the regression approach and the corresponding results are shown in chapter 4, followed by the classification approach in chapter 5. Finally, chapter 6 summarizes the findings and draws conclusions. In addition, an outlook is given what could be considered in future work.

2 Data Pre-processing

The dataset under consideration is the IMDB 5000 dataset which has 5043 instances and 28 variables (from now on just referred to as “dataset”). Each observation corresponds to a movie with details on title, budget and principal actors among others. A movie can be uniquely identified by its IMDb ID called “tconst” (short for title constant). This unique identifier is incorporated in each movie’s unique link to its IMDb page in the original dataset. For pre-processing purposes, the unique identifiers were extracted from the links and added in a new column to the dataset.
At this point, a crucial observation must be made. Although the observations in the dataset are called movies throughout this report, the observations in the dataset have different formats, some of them are TV series for example.
As stated in the introduction, there are a lot of incomplete observations. Missing values are spread over many variables in the dataset. Some variables have a lot of missing values (such as movie gross with 884 missing values) while others only have a few (such as movie color with 19 missing values). To address this issue several techniques were deployed like dropping instances, assigning values based on assumptions and checking values (for example from IMDb itself). For the latter, additional datasets freely provided by IMDb[4] were imported to gather information for some of the missing values.

[4] IMDb (2018b)

Checking the unique identifier of movies, it was found that there was a total of 124 duplicate entries. These duplicates were removed in a first step from the dataset.

2.1 Preparation of the Dataset

This section provides a thorough examination which addresses how variables and missing entries were treated by considering one variable at a time. This will also give a better understanding of the dataset as a whole.

IMDb scores & binned scores

IMDb score is a numeric variable which represents the quality of a movie. It is the dependent variable over which regression analysis was performed in an attempt to explain any relation between IMDb score and the other variables in the dataset.
For the classification setting, the approach of the authors of the Kaggle competition was applied and a new dependent variable was created. The IMDb score scale was divided into four score ranges, called score bins. Then, the movies were assigned a score bin according to their respective IMDb scores. The score bins cover the following ranges: IMDb scores between 1 and 4 (bin 1), 4 and 6 (bin 2), 6 and 8 (bin 3) and 8 and 10 (bin 4). These score ranges can be seen as quality levels of movies. Similar to the categorisation of Saraee et al. (2004), they can be interpreted as “terrible” for bin 1, “poor” for bin 2, “average” for bin 3 and “excellent” for bin 4. Using the same binning approach as the Kaggle authors, the results can be compared to those of the Kaggle competition. Whenever the dependent variable is addressed in the report, it is simply mentioned as the set of binned scores.
In addition, a second set of score bins were created. It was necessary to compute missing values for two variables (the budget and the box-office gross of movies) and divided into the following ranges: 1 - 4 (bin 1), 4 - 5 (bin 2), 5 - 5.5 (bin 3), 5.5 - 6 (bin 4), 6 - 6.5 (bin 5), 6.5 - 7 (bin 6), 7 - 7.5 (bin 7), 7.5 - 8 (bin 8) and 8 - 10 (bin 9). This set of score bins was only used for pre-processing of missing gross and budget values. Whenever this set is addressed in the report, it’s explicitly mentioned as the set for pre-processing.

Genres

This is a pre-existing variable which can contain multiple genres for one movie. Therefore, we created 26 new binary variables (corresponding to the unique genre categories) which denote whether a particular movie belongs to that category. The pre-existing genre category was dropped.

Budget & Gross

These two variables are considered together because they have the largest numbers of missing values. Also, these missing values are dealt in a similar manner in both cases. Since these variables might have a strong relation to the IMDb score, the large number of observations with missing gross or budget should not be dropped. Hence, specialised methods are adopted which use the set of score bins for pre-processing. In both cases, a ‘nested FOR loop’ algorithm runs over all the observations assigning mean gross/budget of movies in the same genre categories and the same score bin. The algorithm works as follows:

Test each observation (where gross/budget value is missing) against all the other observations in the dataset.
For each comparison, count the number of equal genres between the two movies.
If at least two genres match and the two movies have the same binned score, store the index of the movie.
Assign the mean value of gross/budget of the movies which satisfy the condition in step 3.
One of the drawbacks of this approach is the long processing time. Each nested-for loop takes approximately 60 - 80 minutes to assign the values. Yet, it was assumed that the gross/budget mean of similar movies is a better estimate than the overall gross/budget mean of all movies in the dataset. This is why the set of smaller score bins (as opposed to those of the dependent variable) was used for the matching process of the algorithm.

Name of director

There are 102 movies in the dataset which have no director indicated. Searching in the additional IMDb datasets, director names could be added to seven movies. In addition, checking the remaining movies in the additional IMDb datasets showed that many of them had multiple directors assigned. This is possible when every episode of a series is directed by a different director for example. Overall, 2399 different directors appear in the dataset, 876 of which have directed two or more movies listed in the dataset. Therefore, keeping this variable did not seem reasonable from a regression or classification perspective. Instead, a more promising variable was created: It captures the number of appearances of a director in the dataset, i.e. how many movies did the director of a movie direct in the dataset. These values were assigned to each movie and the remaining movies with a missing director name were assigned the value one. This new variable captures any influence of the numbers of movies of a director on the movie score. Finally, the director name variable was dropped.

Name of actors 1, 2 and 3

For each movie, the three starring actors are listed in the dataset. The names of the principal actors specified for each movie in the dataset were handled similarly to the director names. Many different actors appear in the dataset. The actor 1 attribute lists about 2000, the actor 2 attribute about 3000, and the actor 3 attribute about 3500 unique actor names. Many of them appear more than once in the dataset (within each of the variables). With the same reasoning as for the director names, a new variable for each of the actor name attributes was created. These capture the number of appearances of an actor within each actor name variable. Thus, each movie was assigned the number of movies each of its actors are part of in the dataset. Over all actors, there were only a few values missing (documentaries do not have any actors assigned for example); these movies were assigned a value of one. All three actor name features were dropped.

Facebook likes of directors

This variable captures the likes on the Facebook page of a director. Besides 102 movies with missing values (corresponding to the missing values for the director name attribute), 879 directors had zero likes on their Facebook pages. As already mentioned in the example of James Cameron, this is not true for an unknown number of them. Moreover, a certain amount of directors might have never had a Facebook page, especially those who lived and

directed before Facebook went live. In essence, the number of Facebook likes is a measure of popularity. Therefore, a value of zero for nearly one fifth of the movies (including those with missing values) is introducing a large and flawed bias. In order to overcome this problem and reduce the bias, all of these movies were assigned the average number of likes (i.e. popularity) of those movies which had director Facebook likes specified. Certainly, this is only an estimation and an unknown degree of flawed bias remains.

Facebook likes of actors 1, 2 and 3

This variable captures the likes on the Facebook pages of actors 1, 2 and 3 respectively. This variable can be used as a measure of popularity similar to the director Facebook likes. As opposed to directors, there are not many missing and zero values for the actor 1 attribute and the numbers are only slightly increasing for the actor 2 variables. The actor 3 attribute has the most movies with missing (23) and/or zero (89) values. With the same logic as for directors, all missing and zero values were replaced with the mean popularity within each actor Facebook likes variable.

Facebook likes of cast

This variable summarizes the overall popularity of the cast of a movie. Several movies had a value of zero assigned, in that case the cast Facebook likes were estimated by the sum of the Facebook likes of the actors.

Facebook likes of movies

This variable seemed to be interesting because it reflects the popularity of movies (with the same interpretation of Facebook likes outlined for the director Facebook likes attribute). There might certainly be a correlation between the popularity and the quality of a movie. Yet, more than 2000 movies in the dataset had zero Facebook likes. As for director Facebook likes, there might be wrong values assigned to the movies or the movies are very old and, hence, do not have a Facebook page. As before, those values were replaced by the mean of Facebook likes of the movie having a non-zero value in order to reduce the bias.

Number of people featured on movie poster

There were thirteen instances with missing values. These were filled manually checking movie posters available on the IMDb page of each movie. In doing so, it was possible to keep the movies in the dataset.

Color

This is a categorical variable with only two levels: ‘Black & White’ or ‘Color’. There were nineteen movies where the color attribute is missing. As for the number of people featured on movie poster, the information available on IMDb movie pages was used. For all nineteen instances, the movies were colored. It must be noted that more than 95% of the movies in the database are coloured and the variable may not be of much significance.

Content Rating

Content rating of movies is a categorical variable with 18 levels. Content ratings are not consistent among countries and have also been subject to change over time within a single country. There were 303 movies without a content rating indicated. Since these would have had to be checked manually, observations with missing values were dropped. Moreover, the variable had specific categories with a small number of observations. Hence, these were merged with categories of a similar content specification which are more common in the dataset.

Duration

15 movies did not have a duration specified. These values were looked up on the IMDb pages of the movies. The three movies for which the duration could not be found were assigned the mean duration of all movies.

Plot keywords

This variable gives some hints on the plot of the movie. There are numerous unique plot keywords over all movies and each movie has been assigned possibly many of those. A similar approach as for the genres attribute would have resulted in a very large number of new binary variables which might have impacted the analysis of the dataset in a negative way. Therefore, this variable was dropped.

Language & Country

95% of the movies are in English. Since about 75% movies are from the US (with only 18 of those in a different language than English), there is a close connection of the two variables. Therefore, the language attribute was removed. Of the remaining 25% of the movies which are not from the US, the UK provides the second largest number. All other countries are only reflected by a small number of movies. This is why all countries apart from the US and the UK were grouped together into a new country class called “Other”.

Aspect ratio

Although the aspect ratio of a movie does not seem to have a big impact on the movie’s quality, this variable was kept. The two main ratios with the most movies are 1.85 and 2.35. All other ratios only represent small numbers of movies. Therefore, movies having a different aspect ratio than the two main ratios were grouped to a new aspect ratio class called “Other” together with movies having no aspect ratio value specified.

Year when a movie was published

The year when a movie was published might have an impact on the IMDb score[5]. Most of the missing values for this attribute could be found in the additional IMDb datasets. The very few remaining movies with no year value specified were removed from the dataset.

[5] Johnston (2009)

Number of external reviews

IMDb provides links to external reviews on a movie's page. There are 49 movies where these values are missing. This variable may be a critical feature since a movie review addresses the quality of a movie. Thus, these values shouldn't be replaced by some estimator. Otherwise, this may mislead the model if assigned an incorrect value. Moreover, the number of reviews is changing over time. Therefore, checking the movies' IMDb page was not considered.

Number of user reviews

When a user votes for a movie on IMDb, he/she can also give a written review. By the time this feature was addressed for pre-processing, the 21 movies which have missing values for this attribute had already been dropped.

Number of users who voted

The number of users who gave an IMDb score vote did not have any missing values. It was assumed that an IMDb score is only representative if enough votes have been given. Therefore, all movies with less than 500 votes were dropped from the dataset.

Unique identifier, IMDb link & Movie title

These variables are unique to each record and hence are of no use from a regression or a classification perspective. Therefore, they were dropped at the end of the pre-processing process.

All features had been cleaned and stored in a dataset named "IMDB". It contains 4497 movies and 42 predictors as well as the dependent variables for regression (IMDb scores) and classification (binned scores). All numerical variables were checked for skewness. Most of them were highly skewed. Since some classification methods do not work well on such data, log transformation, scaling and centering were applied to all numerical variables which were then stored together with all categorical variables in a second dataset called "IMDBtrans".
In addition, the data was split into a training and a testing set with an 80:20 ratio by random sampling. These sets were then used in the regression and the classification approach.

2.2 Exploratory Analysis on the Training Data

Before the regression and the classification settings were applied on the data, the features were analysed in an exploratory manner. This analysis was conducted on the IMDB training dataset.
First, the distribution of the dependent variable for regression, IMDb scores, and for classification, the binned scores, were plotted[6]. Comparing the binned scores to the IMDb scores, a similar shape is observed.

[6] Refer to appendix 8.2, figure 3

Next, correlation[7] among the numerical variables was checked. A positive significant correlation can be seen among the following pairs of independent variables:
Actor 1/Actor 2 Facebook likes and the cast total Facebook likes: This correlation may be explained by the fact that the actor 1 and 2 are a part of the cast and, hence, constitute a certain proportion of the total cast Facebook likes.
Number of users who reviewed the movie and total number of users who voted for the movie: This correlation may be a result of the fact that the more users vote for a movie, the more also give a review.
This correlation is undesirable and techniques like interaction terms are used to address this issue in regression.
However, any correlation between the dependent and independent variables is desirable. This link is also captured by plotting the numerical variables against the IMDb scores. A closer inspection of variables like number of external as well as user reviews, duration, number of users who voted for a movie against the IMDb score reveals the associated patterns. Surprisingly, the budget of a movie does not seem to have much impact on the IMDb score.

Barplots of the categorical variables subdivided by the binned scores reveal an interesting pattern. All the plots show that the individual categories for each of the 'factor-type' variables follow a distribution similar to the distribution of the binned scores. This phenomenon may hint towards the fact that no distinction can be made between the different categories of a variable. Yet, it's too early to draw an inference.

3 Performance Measures

3.1 Regression

There are various performance measures that enable us to judge the accuracy of the Regression model:

R^2 tells us the proportion of variability of Y that is explained by the current X's in the model. Its Formula is as follows:

$$R^2 = \frac{Explained\ Sum\ of\ Squares\ (ESS)}{Total\ Sum\ of\ Squares\ (TSS)} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\hat{y}_i = y_i - \varepsilon$.
Note that the R^2 always increases as more variables are added to the model, even if they are not related to the response. This is because, whenever you add a new variable to the model, the value of its coefficient can be zero, in which case proportion of explained variance (R^2) remains unchanged or take a non-zero value because it improves the quality of fit. Thus, R^2 can never be smaller by adding a new variable.

Adjusted R^2 provides a remedy for this as it penalises the addition of new variables, unless they have an impact on Y.

[7] Correlation Plot in the Appendix 8.2, figure 4

RSE is an estimate for the standard deviation of ε $(the\ irreducible\ error)$ i.e. the average amount that the Response (Y) deviates from the regression line.
For the above model, the RSE for a particular IMDB score varies by approximately 0.8 points. Furthermore, Adjusted R^2 is relatively low at just 49%. This may be due to several factors such as non-linearity between the response and (some) predictors in the data, multicollinearity (correlation between several predictors), non-normality of the data etc. These issues will be tackled later.

The **F-Statistic** and p-values are both used to test the significance of the predictor variables in the model i.e. it answers the question of whether there is a relationship between the response and predictors. The F-statistic is constructed under the Null Hypothesis

H_o: None of the Predictors (β_j) are significant

versus the alternative hypothesis

H_a: at least one β_j is non-zero.

A large F-Statistic would indicate a significant relationship between the response and predictors, whereas if there is no relationship, the F-stat would take a value close to one[8].

The **p-value** indicates the **risk** of rejecting the null hypothesis. A small p-value would indicate a small risk of rejecting H_0 and thus it will be rejected. A large p-value would indicate a large risk if H_0 was to be rejected and thus we do not reject it. For the Multiple Linear Regression model, there is a very high F-stat and an extremely small p-value which means we can reject the Null hypothesis and conclude that there is significant evidence of a relationship between the response and the predictors.

3.2 Classification

In order to assess the performance of a particular classifier in terms of categorising the different classes, we need some metrics. These metrics are called performance measures. There are several ways to evaluate the performance, but only a few are relevant given the context of the problem which involves multi-category classification. Given the task of categorising the movies into different score bins, it is adequate to use the following measures which would evaluate the performance of the different classifiers overall (like accuracy, kappa) as well as evaluate their performance in terms of the individual classes (like sensitivity).

Accuracy

It is the ratio of correctly classified observations to all the observations. It is one of the most commonly used performance measure however, it is recommended to use it when the data is symmetric i.e. observations are symmetrically distributed amongst the different classes. The penalty/loss from misclassification must be the same in both the cases i.e. when the classifier produces a false positive or a false negative.
However, in the dataset used, most of the classes are imbalanced and hence, accuracy is probably not the best way to access the model's efficiency. This leads to the introduction of statistical measure called Cohen's Kappa.

Cohen's Kappa (κ)[9]

This statistic is computed as follows:

[8] James et al. (2017 p. 76)
[9] Columbia University (2018) and McHugh (2012)

$$\frac{(observed\ accuracy\ -\ expected\ accuracy)}{(1\ -\ expected\ accuracy)}$$

where expected accuracy can be calculated from the performance matrix as follows:

$$\sum_{i} \frac{(Conditional\ positive\ +\ Predicted\ conditional\ positive)}{(total\ number\ of\ observations)^2}\ , i \in Classes$$

It must be noted that this statistic is always used in conjunction with a confusion matrix and it assumes values between 0 & 1. The kappa statistic can be majorly used in two ways:
It can be used to measure the efficiency of the classifier model, especially when the different classes are highly imbalanced. It is more reliable than accuracy because accuracy doesn't take into account the skewness of the class distribution, however this skewness is accounted for in the expected accuracy.
It can also be used to measure which classifier model performs the best (i.e. testing a model against a random classifier). The model with the greatest kappa statistic outperforms all the others. Generally, a kappa greater than 0.4 indicates a moderate performance, greater than 0.6 indicates substantial/good performance while anything above 0.8 is seen as an excellent performance.
Since the data distribution is imbalanced amongst the four classes (in binned score variable), kappa is good way to access the classifier performance,
However, there may be times when the one performance measure maybe more important than the other. For instance, in a particular instance it may be more desirable to use sensitivity of each class or maybe it takes too long to train the data for a task which is time bound. Hence, we at times may need to look at measures in combination.

Computational Efficiency[10]

Computational Efficiency is the amount of resources i.e. the amount of time (specifically in our case) used by an algorithm. To measure the performance of the different classifier, it can be interesting measure especially when the tasks are time bound or when longer computation time results in wastage of other resources (for instance, paid hours). In the different classifier, we will only use the time required by the method to train the model as the criterion for evaluation and not include the time required for pre-processing the data as the pre-processed data is freely available and commonly utilised by each classification method.

Sensitivity

Sensitivity evaluates the performance of the classifier with respect to each class. It detects classification rate amongst the true class i.e. it gives the probability of correctly classifying an observation in a particular class given it truly belongs to that class.

[10] IGI Global (2018)

Precision

Sensitivity evaluates the performance of the classifier in terms of each class. It detects classification rate amongst the predicted class i.e. it gives the probability of correctly classifying an observation in a particular class given it was predicted to belongs in that class.

4 Regression

In this section Linear Regression Analysis is performed on the cleaned IMDB dataset in order to make inferences on how the Response Variable "imdb_score" is affected by the various Independent Variables in the dataset. In doing so, the following questions will be addressed:

- Which features have a significant impact on IMDB Score?
- What is the effect of each feature on IMDB Score?
- Given a set of features, can IMDB Score be predicted?

Finally, some of the drawbacks of linear regression will be highlighted, along with remedial measures for (some of) them.

4.1 Simple & Multiple Linear Regression

Linear Regression is the case of modelling a Linear relationship between a scalar, **dependent** variable (Y) and at least one **independent/explanatory** variable (X).
The case of one explanatory variable is called **simple linear regression**[11].

The case where multiple explanatory variables are used to model the relationship with the Response variable is called **multiple linear regression(MLR)**.

Performing a Regression of IMDB Score on all the variables in the dataset gives us the following results:

Performance Measures of the Multiple Linear Regression Model	
Residual Standard Error (RSE)	0.7888
(Multiple) R-Squared	0.4901
Adjusted R-Squared	0.4849
F-Statistic	87.54
p-value (for entire model)	<2e-16

Table 1: Results for MLR of IMDB Score on all the other Predictor Variables

4.2 Estimating & Interpreting Model Coefficients

Running the aforementioned Simple Linear Regression model in R gives the following output:

[11] Refer to the figure 2, appendix

Coefficients:				
	Estimate	Std. Error	t value	Pr(> |t|)
(Intercept)	6.150	1.743e-02	352.88	<2e-16 ***
Num_voted_users	3.383e-06	1.035e-07	32.69	<2e-16 ***

Table 2: Coefficient Estimates for Regressing IMDB Score on the "num_voted_users" variable

The **Intercept** term (β_0) indicates that the mean IMDB Score for a movie is 6.15.
The **Slope** term (β_1) is the value of Y when X changes by one unit; in this case, when the Number of voted users increases by 1, the IMDB Score for a movie will rise by 0.0000034 units.
Thus, the IMDB Score for a movie which has only one predictor variable ("num_voted_users") is approximately **6.15**.

4.3 Issues & Improvements to the Regression Model

In this section, the Issues ingrained in a Linear Regression Model will be discussed, along with remedial measures for (some of) them.

Multicollinearity

Multicollinearity is the situation where one or more predictors in the Model are related not just to the Response, but to each other. Whereas slight collinearity isn't necessarily a problem, strong collinearity between variables will inflate the standard errors of the regression coefficients and make the model unstable. This can cause variables which are highly related to the response to be deemed insignificant.
A way to detect multicollinearity is to compute the **variance inflation factor (VIF)** which is simply:

$$VIF(\beta_j) = \frac{Var(\beta_j)\ fitting\ full\ Model}{Var(\beta_j)\ fitting\ Model\ with\ \beta_j\ Only}$$

As a general rule, if a variable has a VIF greater than 5, this usually indicates multicollinearity[12].

Variables with VIF Above 5 before Adj. for Multicollinearity	**Variables with VIF Above 5 after Adj. for Multicollinearity**
Actor_1_facebook_likes = **219.14**	-
Actor_2_facebook_likes = **19.29**	-
Cast_total_facebook_likes = **325.01**	**2.07**

Table 3: Left: Variance Inflation Factor (VIF) values for VIFs greater than 5 in the Multiple Regression Model. Right: Removing the least correlated variables with the Response, it can be seen that the VIF for one of the highly collinear variables is within the acceptable range (of below 5).

How do we decide which variable to exclude?

Upon introspection of the correlation with each of the highly collinear variables on IMDB Score, there is a similar correlation amongst them, with "actor_1_facebook_likes" and "actor_2_facebook_likes" being the least correlated. However, regressing IMDB Score by taking out each of the 3 collinear variables, it is found that the model without

[12] James et al. (2017 p. 101)

"actor_2_facebook_likes" gives us the highest Adjusted R^2 value of 48.43%. Thus, the "actor_2_facebook_likes" variable will be excluded from the IMDB dataset. By further removing the next least correlated variable with the response, "actor_1_facebook_likes", and calculating VIF scores, it is observed that no variable has a VIF of 5 or above.

Relaxing the Additive Assumption

Linear Regression has the restrictive assumption that the change in the Response to a unit change in X_j is independent of the other predictors. This certainly may not be the case as IMDB Score may also be responsive to a simultaneous change in 2 or more variables.
Looking at the data, two such relationships can be observed:
If the "**cast_total_facebook_likes**" variable changes, then "**movie_facebook_likes**" may also change, thus affecting IMDB Score.
Further, if "**num_critic_for_reviews**" changes, then "**num_voted_users**" and "**num_user_for_reviews**" will also change, also affecting IMDB Score.

These extra predictors are called **interaction terms**. For illustrative purposes, the procedure of adding only one of the above terms to the regression model will be demonstrated:
Suppose X_1 = cast_total_facebook_likes
X_2 = movie_facebook_likes

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \varepsilon \\ &= \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \varepsilon \\ &= \beta_0 + \boldsymbol{\beta'}_1 X_1 + \beta_2 X_2 + \varepsilon \end{aligned}$$

How does this relax the additive assumption?
Since $\boldsymbol{\beta'}_1$ changes with X_2, the effect of X_1 on Y now also depends on X_2 and thus is no longer constant.

Adding the above two interaction terms to the regression model and running the summary () function on it gives an improved Adjusted R^2 of **49.34%** (up from 48.43%).

Non-Linearity Non-Normality & Heteroscedasticity in Data

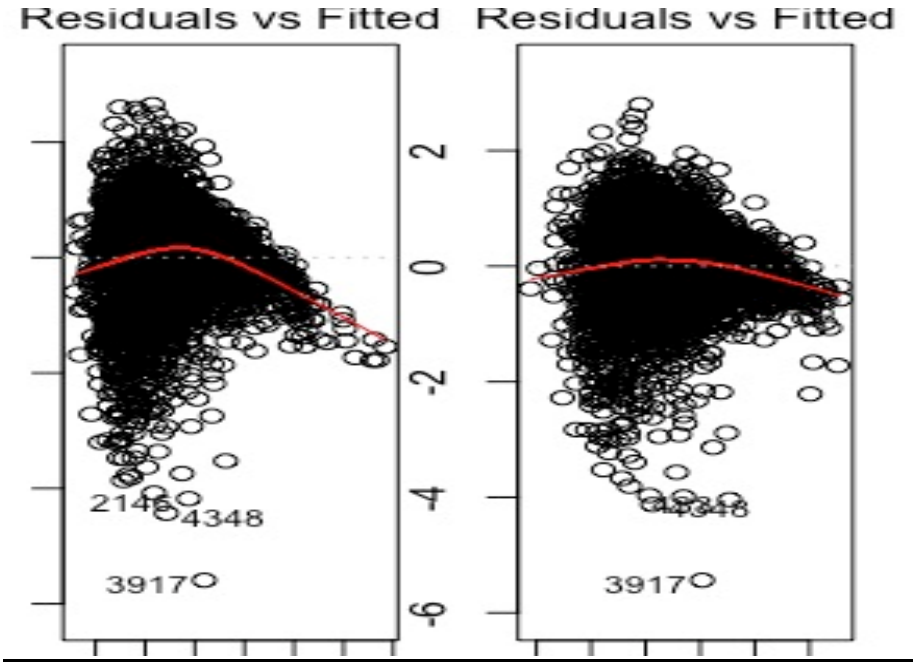


Figure 1: Plot of Residuals VS Fitted Values for the IMDB dataset with the interactive terms, before (left) and after (right) the inclusion of Polynomial terms. The blue line is a smooth fit to the residuals, making it easier to identify a trend.

Non-Linearity:

The linear Regression model assumes a linear relationship between the predictors and the response. However, if the true relationship is not linear, the predictive accuracy of the model can be significantly reduced.

The plot on the left shows if residuals have a non-linear trend. If the residuals are randomly spread around a horizontal line (without a distinct pattern), this means that there is no non-linear relationship. However, in the above diagram, a parabolic relationship in the residuals on the left can be seen. By including **Polynomial** terms (of up to degree 3 for example) in the regression model, this defect can be alleviated.

However, before adding polynomial terms, the data needs to **centered** to prevent the polynomial terms being highly correlated with each other. The **scale ()** function in R is used for this and the centered dataset is named as **IMDBcenter**.

Polynomial terms for the (significant) variables "duration", "**num_voted_users**", "**num_critic_for_reviews**", "**num_user_for_reviews**", "**title_year**", "**movie_facebook_likes**" and "**gross**" are then included. Adjusted R^2 rises from 49.34% to **55.42%** indicating a significant improvement in the model fit. The plots in Figure 2 reflect the distribution of residuals before and after the addition of polynomial terms.

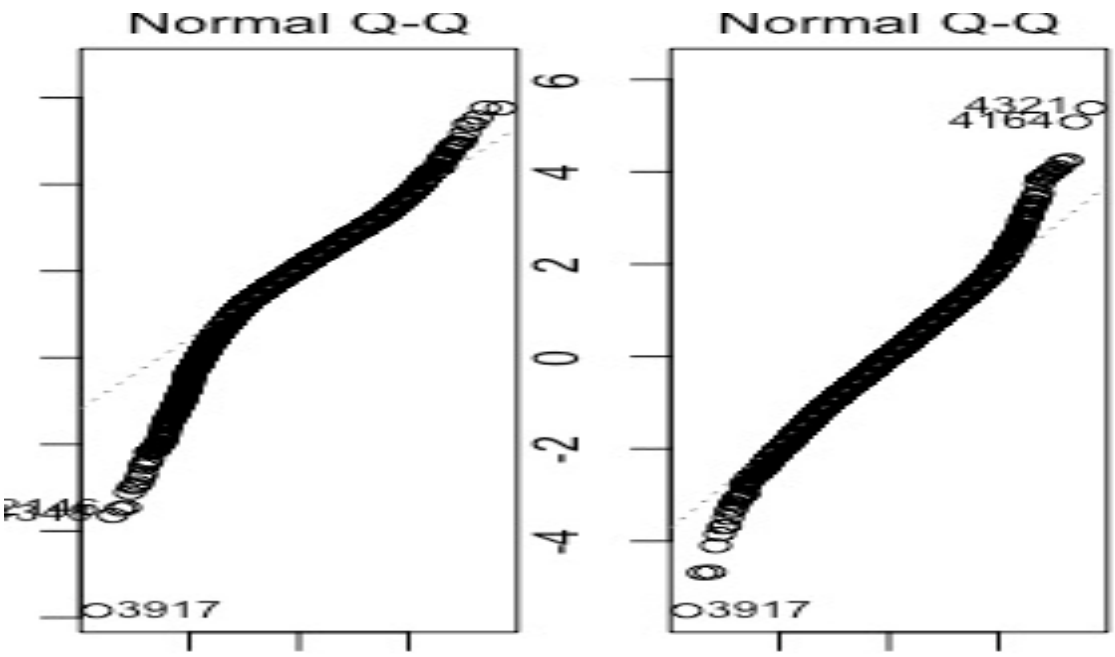


Figure 2: Normal Q-Q plots of the residuals before (left) and after (right) the Box Cox transform; if the data stray away from the dotted-line, this is an indication that the data doesn't follow a normal distribution. In the plot on the right, it is clearly visible that the data more closely follows the dotted line as compared to the plot on the left.

Non-Normality and Heteroscedasticity:

Another restrictive assumption of linear regression is that it assumes the Residuals follow a Gaussian Normal Distribution i.e. $\varepsilon_i \sim N(0, \sigma)$. From the Normal Q-Q plot on the right in Figure 2, it can be seen that the residuals deviate quite severely from the straight line indicating Non-Normality in the data.

Linear Regression also assumes that the error terms have a constant variance (Homoscedasticity) i.e. $Var(\varepsilon_i) = \sigma^2$. In reality, the variance of the error terms is not constant (Heteroscedasticity).
See the plots[13] for evidence of how the Box Cox transform has addressed these two issues.

To address these issues, the **Box Cox Transform** can be used for the Response variable.
It was developed by statisticians George Box and David Cox in 1964 "to transform non-normal dependent variables into a Normal Shape".
It uses an **exponent** lambda which varies from -5 to 5; all the values of lambda are considered and the "optimal" value is selected via the method of **log-likelihood** where "lambda is chosen to maximize the profile log likelihood of a linear model fitted to data"[14].
This results in a model which represents the closest approximation to a normal distribution curve. The formula used is as follows:

$$Y = \begin{cases} \dfrac{y^{\lambda} - 1}{\lambda} & if\ \lambda \neq 0 \\ \log y & if\ \lambda = 0 \end{cases}$$

Note that this formula only works for positive data. Since, IMDB score is a non-negative variable, the case where Box Cox can be applied to negative y-values will not be considered.

[13] Refer to appendix 8.2, figure 6
[14] Galili (2016)

The **boxcox(lm.model, ..)** command in R can be used to find the optimal value of lambda that maximises log-likelihood. It is found that this value is approximately 3. (Refer to appendix)[15]
After transforming the response using the optimal value of lambda, the Adjusted R^2 rises from 57.12% to **64.06%**. The Quantile plots of residuals[16] before and after the transformation of the Response using Box Cox transform.

Does Box Cox Always Work?

The Box Cox Transform isn't a guarantee for Normality. It assumes that among all transformations with Lambda values between -5 and +5, transformed data has the highest likelihood (although not a guarantee) to be normally distributed when **standard deviation is the smallest**. This means it actually checks which values of lambda give the smallest standard deviation of the Response. So, in order to be sure that the transformed data is (approximately) normal, a **probability plot** needs to be used. This can be done in R using the **qqnorm ()** and **qqline ()** commands (Refer to **appendix** to view plot in R).

4.4 Making predictions From the Linear Model

Now that most of the issues pertaining to the linear model have been alleviated, we can make predictions on the response. To do this, a random split of **80:20** between the training and test sets is used. Doing so will enable us to train the model to predict IMDB Score on 80% of the data and then test the model on the other 20% which it has not seen before.

Calculating Prediction Accuracy and Error Rate:

A correlation between the actual and predicted values of IMDB Score can be performed as a simple accuracy measure. Doing so gives a relatively high correlation of **79.23%**.

In order to calculate the prediction accuracy of our model, the MinMax Accuracy & MeanAbsolutePercentageError (MAPE) statistics can be used:

$$\text{MinMaxAccuracy} = mean\left(\frac{\min(actual, predicted}{\max(actual, predicted)}\right)$$

$$\text{MeanAbsouletPercentageError (MAPE)} \ = mean\left(\frac{abs(actual - predicted)}{actual}\right)$$

By observing the formulas for the above two statistics, it is quite easy to understand how they work. Suppose we have a data frame of 2 columns: the actual and predicted values of IMDB Score for each record.
The MinMax Statistic takes the minimum value of each row and divides it by the maximum. The better the prediction, the higher the value, with perfect prediction having a value of 1.
The MinMax Accuracy of the model is approximately **79%** and the MAPE is approximately **42%.**

[15] Refer to appendix 8.2, figure 7
[16] Refer to appendix 8.2, figure 8

5 Classification

In addition to the regression analysis, several classification methods were applied. Whereas the regression provided insights into inference, this chapter analyses how well the quality of a movie can be predicted. This is particularly interesting since the results can be compared to the previous analysis of the same dataset which was mentioned in the introduction.
The analysis comprises the following classifiers: Multinomial Logistic Regression, Linear Discriminant Analysis (LDA), k-Nearest Neighbors (kNN), linear, radial as well as polynomial Support Vector Machines (SVM), Random Forests and Boosting.

5.1 Model Tuning and Training

First, the models of the algorithms were tuned and/or trained on the training dataset. Three times repeated 10-fold cross validation (CV) was applied on each training process, and the kappa statistic was used as the universal tuning measure. Because of very long processing times and heavy computations of the tuning processes, CV was not included here.

Multinomial Logistic Regression

Multinomial Logistic Regression predicts probabilities of an instance for each class. It is used when there are more than two classes for the dependent variable. In a sense, this method is a further development of logistic regression: a class is assigned to an observation using One vs One (OvO) classification. This method compares probabilities for all classes one by one by training multiple logistic regression models. Observations are assigned to a class using the majority vote. OvO may not be the most ideal method to make probabilistic predictions because it makes assumption of independent classes within the dependent variable[17].
Also, it doesn't make the assumption of normality and the variables don't need scaling as it can be accounted for in the regression coefficients. Hence, we use the pre-processed IMDb dataset without any scaling or transformation.
However, in R, Multinomial Logistic Regression uses the slightly different One vs All (OvA) classification in conjunction with the softmax function and assigns the instance to the model with the highest output[18]. Each model corresponds to a class of the dependent variable when using OvA. This is a good choice since our objective is making predictions using probabilities and multinom() classifies using OvA classification method which doesn't make the IIA assumption.

Support Vector Machines

Support Vector Machine is a classification approach which is a more flexible than support vector classifier (used for linear boundaries). It is used when the decision boundary is non-linear or not perfectly linear. However, this method is used for binary classification. In order

[17] Belsley (1991)
[18] Polamuri (2017)

to extend its application - to the case of more than two classes - it is integrated with the OvA or OvO approach.
For all the different SVM methods, we use the transformed dataset because the data needs to be scaled to 'avoid attributes in greater numeric ranges dominating those in smaller numeric ranges'[19].
The motivation to use these methods comes from their ability to capture the nonlinearity when the Bayes decision boundary is not linear.

Linear Kernel

The Linear Kernel is used for cases where classes are separable by a hyperplane. The bayes decision boundary may not be completely linear but smooth enough to fit a hyperplane which separates the classes with as few violations/penalties as possible.
In this case, every misclassification (on the training data) costs 0.75.

Polynomial Kernel

The Polynomial Kernel is used for cases where the classes are separable by a polynomial-like curve. The methods are tuned to work best with a polynomial of degree 2 where scaling is done by a factor of 0.001. Misclassifications (on training data) are penalised at a cost of 300.

Radial Kernel

The Radial Kernel is used for cases where the decision boundary is highly non-linear. The motivation to use this method is the indication of non-linearity since SVM performs better with a polynomial kernel instead of the linear kernel. The parameters are used to penalise misclassifications at a cost of 10 and the other parameter (sigma/lambda) takes the value 0.01

The methods discussed beyond this point are capable to handle multi-class classification and don't have to rely on OvO or OvA methods.

[19] Hsu et al. (2016)

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is implemented because it can be easily used to make a multi-class predictive classifier. It assumes that each predictor follows a Gaussian distribution. Also, it is influenced by the magnitude of the variables. Hence, classification method was trained on the transformed dataset. The transformed dataset is log transformed to get rid of the skewness. It is then scaled to have standardized values.

k-Nearest Neighbors

k-Nearest Neighbours predicts the probabilities of an instance for each class based on the class of the nearest k neighbours. It calculates the probability of an instance belonging to class as the number of members of that class among the k nearest neighbours divided by k. It assigns observation to the class with the highest probability. It uses the Euclidean distance to determine the neighbours and calculates the distance between observations by making the assumption that the predictors are independent.
It must be noted that this method only works with numeric variables and hence information from factor type variables maybe lost. For this reason, variable selection was done using principal component analysis (PCA). PCA create new variables or projection axis which can explain most of the variance in the data. This is an attempt to save the classifier from the curse of dimensionality[20] by trying to use as few as predictors as possible (which also capture most of the variability in the data). In particular, the classifier only uses 27 out of the 46 available variables and assigns class based on 21 nearest neighbours (i.e. k = 21). The classifier uses the transformed dataset as decisions are influenced by the magnitude of the variables.
It must be noted that kNN is an exceptional case where there is no training. Also, the testing set can be seen as a way to validate the

Random Forests (or Bagging)

Random Forest classifier is a tree-based method and a special case of bagging where the number of predictors considered at each decision node is equal to the square root of the total number of predictors available. It helps decorrelate trees grown since these variables are selected randomly.
There is no need to cross validate the model as the trees are trained over bootstrap sample and models are validated using out-of-bag (OOB) observations. The other parameters considered while implementing this method include the number of trees to be grown. It must be noted though that in the train and tuning of the model the method works best when mtry=39 i.e. at decision node the model chooses from 39 random variables out of the available 42 variables. This is indicating that the method tends to perform better as on a normal bagging method instead of random forest.
The method is tuned and trained using the pre-processed data set with no transformation as it makes no assumption of normality and the decisions are not affected by magnitude of variables.

[20] James et al. (2017 pp. 59-126)

Boosting

Boosting is also a tree-based method, but it is a slow learning method. It involves a learning parameter called shrinkage (learning rate) which builds a new tree using the information from the old one. However, there is significant difference between bagging and boosting. Here the number of trees required to make a good predictive model depends on the value of shrinkage. There exists an inverse relation between the number of trees and the learning rate. The other parameters involved are the interaction depth and the number of observations in each leaf node. The former controls the maximum number of decision nodes present when growing a tree, while the latter acts as indicator when to stop growing a tree - the tree stops growing when the number of observations in each leaf node is less than or equal to this number.
In this case, the classifier was trained from 300 trees with a maximum of 7 decision nodes and 5 observations in each leaf node. It learnt at a rate of 0.05 from each of the trees.
The method is tuned and trained using the pre-processed data set with no transformation as it makes no assumption of normality and the decisions are not affected by magnitude of variables.

5.2 Application and Performance on Testing Dataset

After tuning and/or training, the score bins for the movies in the test dataset were predicted using the models after tuning and/or training and compared to the true classes. All the classification methods described above were evaluated using the performance measures discussed earlier. They are evaluated in the following three categories:

- The overall performance - Accuracy and Cohen's Kappa
- Class-wise Performance - Sensitivity and Precision
- Computational Efficiency - Tuning, Training and Testing Time

The corresponding confusion matrices can be found in the appendix.

Overall Performance

The overall performance is evaluated using the performance measures: Accuracy and Kappa.

Overall Performance Measures (in percentage)		
Classifier	Accuracy	Cohen's Kappa
Multinomial Logistic Regression	73.11	38.61
Linear Discriminant Analysis	73.78	44.07
K- Nearest Neighbours	71.22	34.21
Support Vector Machines – Linear Kernel	77.00	52.43
Support Vector Machines – Polynomial Kernel	76.67	51.10
Support Vector Machines – Radial Kernel	77.67	55.26
Random Forests	79.98	55.72
Boosting	82.33	62.28

Table 4: Reviewing performance measures: Accuracy, Cohen's Kappa, Sensitivity and Precision

As can be seen in Table 4, Boosting outperforms every other classifier. This can be attributed to its slow learning rate and the highly efficient and technical way the pre-processing of the data is handled. The achieved accuracy is up to 5% better than the benchmark set by the authors of the Kaggle competition. Their highest test accuracy of 76.58% was achieved with Random Forests. The same classifier yielded an accuracy of 79.98% in this study. The three different SVMs performed equally well in terms of accuracy (beating the benchmark as well), this is rather surprising since they are assumed to work well on different types of decision boundaries. Comparing the classifiers in terms of the Kappa performance measure, again Boosting clearly outperforms all the other methods, followed by Random Forests. The Kappa values for the SVM classifiers differ as opposed to the accuracy statistics.
In this study, kNN under performs in comparison to the benchmark study [Accuracy of 74.56%]. This can be a result of the principal components which are not able to effectively capture the variance in the given data.
It would be interesting to see if the results improve for logistic regression with some kind of feature selection analysis.

Class-wise Performance

The class-wise performance is evaluated using the performance measures: Precision and Sensitivity.

Class-wise Performance Measures (in percentage)					
Classifier	Measure	Bin 1	Bin 2	Bin 3	Bin 4
Multinomial Logistic Regression	Sensitivity	0	44.84	91.39	35.89
	Precision	NA	64.20	75.21	77.78
Linear Discriminant Analysis	Sensitivity	17.88	52.38	87.78	43.59
	Precision	29.41	63.77	78.70	60.71
K- Nearest Neighbours	Sensitivity	3.57	48.02	89.33	0
	Precision	50.00	60.20	89.33	0
Support Vector Machines – Linear Kernel	Sensitivity	0	63.81	89.22	50.00
	Precision	0	66.67	80.80	88.89
Support Vector Machines – Polynomial Kernel	Sensitivity	0	62.70	87.61	58.77
	Precision	0	66.95	81.57	63.89
Support Vector Machines – Radial Kernel	Sensitivity	68.69	67.70	87.46	58.33
	Precision	22.22	66.16	83.33	82.35
Random Forests	Sensitivity	14.28	61.51	92.94	48.79
	Precision	1.00	73.11	81.33	95.00
Boosting	Sensitivity	21.42	67.06	93.63	56.41
	Precision	50.00	77.17	84.47	88.00

Table 5: Performance Measure: Sensitivity and Precision

As it can be seen in Table 5, the SVM radial kernel performs well in predicting all the binned score categories in terms of sensitivity. It must be noted that the method which performs best overall, i.e. boosting, is not the best in terms of balanced class-wise performance. It seems that there might be a tradeoff between overall performance and the prediction of bins containing relatively few movies.

Since most of the observations belong to bin 3, it is observed that all the classifiers perform equally well in terms of both sensitivity and precision when it comes to bin 3. However, their performance in bin categories with lower number of movies is generally lower and highly variable (for example in bin 1, sensitivity covers a range from 0 to 68.69) among the methods. While kNN performs very bad in predictions related to bin 4, other methods like SVM linear as well as polynomial kernel perform bad on bin 1, i.e. the other side of the range. None of these methods performed well on both extremities.
When it comes to precision, again there is a method with a more balanced performance over the bins, namely Random Forests, compared to Boosting. However, their performances in this respect is almost the same except in bin 1 where random forest has a perfect score. The similarity in the performance measure might be attributed to the family of classification methods they come from i.e. tree-based methods.

Computational Efficiency

It is important to note at this point that all the values given in the table below correspond to computations which were run on 1.8 GHz Intel Core i5 processors with 8GB 1600MHz DDR3 RAM. These facts are essential since they limit the computation power and speed which was already mentioned before.

Computation Efficiency			
Classifier	Tune Time	Train Time	Test Time
Multinomial Logistic Regression	-	2.98 min	0.05 sec
Linear Discriminant Analysis	-	5.75 sec	0.04 sec
K- Nearest Neighbours	1.17 min	4.81 sec	
Support Vector Machines – Linear Kernel	4.66 min	33.73 sec	0.02 sec
Support Vector Machines – Polynomial Kernel	1.24 hr	1.03 min	0.15 sec
Support Vector Machines – Radial Kernel	39.99 min	1.23 min	0.44 sec
Random Forests	4.02 min	1.99 min	1.79 sec
Boosting	4.42 hr	9.14 min	0.12 sec

Table 6: Performance Measure: Computational Efficiency

The method which took the longest to tune is Boosting. It took around 4 hours to tune the model while k-Nearest Neighbours only took 1.17 min. But this is not a fair comparison given a broad variety of tuning parameters provided to each method. Boosting tried 180 combinations before coming up with the best tuning parameter while kNN just tried 10. Hence, when it comes to measuring the tuning times it makes more sense to compare the tuning times per combination. In this light, it seems that both Boosting and SVM which are slow learning methods didn't take much longer to tune compared to other methods.
The higher training times can be attributed to cross validation which was applied while training methods. However, it is interesting to note that SVM methods trained the models quicker than expected (in comparison to the tuning times per combination) even though the training models are cross-validated. The training time of 9 minutes for the boosting classifier can be attributed to the shrinkage/learning rate of just 0.05.
Finally, it was observed that testing times were very short. Therefore, they were not important in the evaluation of the computational performance.

6 Conclusion

6.1 Interpretation of Results

First of all, the IMDB 5000 dataset is a rather difficult one. Coming back to the limitations stated in the introduction, a lot of time and effort was spent on data cleaning. Estimation of the many missing values and the fact that the dataset includes wrong values make it difficult to come up with valid results. Having gross and budget in mixed currencies is not recognised by data mining techniques which evaluate them wrongly. Updating the dataset to reduce the number of missing and incorrect values could lead to much better results.
The regression analysis showed some interesting findings regarding the relationship between IMDB score and the predictors:

- The **budget** variable is not significant in determining IMDB Score i.e. the quality of a movie doesn't seem to be directly affected by how much is spent to make it. Similarly, and perhaps rather surprisingly, **cast_total_facebook_likes** (an indicator of how many people like the movie cast on Facebook) does not have a significant impact in determining IMDB Score either.
- Although **movie_facebook_likes** (the number of likes on the movie Facebook page) is significant in determining IMDB Score, the interaction between **cast_total_facebook_likes** and **movie_facebook_likes** is not significant in determining IMDB score i.e. a simultaneous change in both variables does not impact IMDB Score much.
- **title_year** surprisingly has a significant impact on IMDB Score; it can be observed that the correlation between the two variables is negative. This suggests that **the older the movie, the better the IMDB Score**. Furthermore, adding the cubic polynomial of **title_year** is not significant, but the quadratic polynomial is significant (i.e. it leads to an improvement in the model fit & hence, in predicting IMDB Score). This suggests that although adding non-linear transforms to the data is important in order to linearize it, it may not always lead to an improvement in the model fit.

Yet, the regression setting that was applied also has some drawbacks. As far as prediction is concerned, there may have been better methods in order to identify which features to omit from the model that do not have a significant impact on IMDB Score for example, **Ridge** and **Lasso** Regression. In addition, **overfitting** was not considered; this is where the model responds too well to the random noise in the data rather than focus on the relationships between variables. It occurs when the model is too complex and can produce incorrect R^2 values, regression coefficients and p-values.

In the classification setting, it was possible to beat the benchmark of the authors of the Kaggle competition mentioned. It turned out that Boosting performed best in terms of Cohen's Kappa as well as accuracy. Other methods such as radial SVM were able to come up with slightly worse overall but more balanced results. One drawback of the Boosting approach was its long computational time for tuning. Random Forests performed second best regarding Kappa and accuracy but only needed several minutes for tuning.

Overall, it was rather surprising that the classifiers would perform that well in predicting quality levels of movies since they were tuned and/or trained on a dataset of bad quality.

The binning of movies was adopted from the Kaggle competition for comparison with the benchmark. It would be interesting to see what happens to prediction performance when the bin ranges are changed. Intuitively, the broader the ranges of bins are (i.e. less bins), the easier it might be to predict the correct bin.
The major shortcoming of the classification setting lies in the choice of predictors. For answering whether a movie will be of good quality before it is released some of the variables which were used for classification (and for regression) would not be available such as number of users who voted for a movie. Therefore, this analysis is not valid for real-world application. Further work could elaborate on this and remove predictors that are not available before a movie is released.

7 References

Baugher, D. & Ramos, C. (2016). The Cross-Platform Consistency of Online User Movie Ratings. Atlantic Marketing Journal. Vol. 5 No. 3 , Article 9.

Belsley, D. A. (1991). Conditioning diagnostics: Collinearity and weak data in regression (No. 519.536 B452). Wiley.

Columbia University (2018). How to Calculate Kappa. Available under http://epiville.ccnmtl.columbia.edu/popup/how_to_calculate_kappa.html (last accessed: 23/04/2018).

Galili, T. (2016). Lambda (Box-Cox transformation parameter) value for forecasting using ARIMA. Available under https://www.r-bloggers.com/lambda-box-cox-transformation-parameter-value-for-forecasting-using-arima/ (last accessed: 23/04/2018).

Hsu, C. W., Chang, C. C., & Lin, C. J. (2016). A practical guide to support vector classification. Available under https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf (last accessed: 23/04/2018).

Hsu, P. Y., Shen, Y. H., & Xie, X. A. (2014). Predicting movies user ratings with imdb attributes. In International Conference on Rough Sets and Knowledge Technology (pp. 444-453). Springer, Cham.

IGI Global (2018). What is Computational Efficiency. Available under http://epiville.ccnmtl.columbia.edu/popup/how_to_calculate_kappa.html (last accessed: 23/04/2018).

IMDb (2018a). FAQ for IMDb Ratings. Available under https://help.imdb.com/article/imdb/track-movies-tv/faq-for-imdb-ratings/G67Y87TFYYP6TWAV (last accessed: 23/04/2018).

IMDb (2018b). IMDb Datasets. Available under https://www.imdb.com/interfaces/ (last accessed: 23/04/2018).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An Introduction to Statistical Learning (Vol. 112). New York: Springer.

Johnston, N. (2009). IMDb Movie Ratings Over the Years. Available under http://www.njohnston.ca/2009/10/imdb-movie-ratings-over-the-years/ (last accessed: 23/04/2018).

Kaggle (2018). Analyze IMDB score with data mining algorithms. Available under https://www.kaggle.com/carolzhangdc/analyze-imdb-score-with-data-mining-algorithms/notebook (last accessed 23/04/2018).

Li, P. (2005). Box-Cox Transformations: An Overview. Available under http://avesbiodiv.mncn.csic.es/estadistica/curso2011/boxcox.pdf (last accessed: 23/04/2018).

McHugh, M. L. (2012). Interrater reliability: the kappa statistic. Biochemia Medica, 22(3), 276–282.

Minitab Blog (2013). Enough Is Enough! Handling Multicollinearity in Regression Analysis. Available under http://blog.minitab.com/blog/understanding-statistics/handling-multicollinearity-in-regression-analysis (last accessed: 23/04/2018).

Oghina, A., Breuss, M., Tsagkias, M. & de Rijke, M. (2012). Predicting IMDB Movie Ratings Using Social Media. 503-507.

Polamuri, S. (2017). How the multinomial logistic regression model works. Available under http://dataaspirant.com/2017/03/14/multinomial-logistic-regression-model-works-machine-learning/ (last accessed: 23/04/2018).

Saraee, M., White, S., & Eccleston, J. (2004). A data mining approach to analysis and prediction of movie ratings. Transactions of the Wessex Institute, 343-352.

Kim, B. (2015). Understanding Diagnostic Plots for Linear Regression Analysis. Available under http://data.library.virginia.edu/diagnostic-plots/ (last accessed: 23/04/2018).

Wired (2017). You should ignore film ratings on IMDb and Rotten Tomatoes. Available under http://www.wired.co.uk/article/which-film-ranking-site-should-i-trust-rotten-tomatoes-imdb-metacritic (last accessed: 23/04/2018).

8 Appendices

8.1 Tables

Variable	Variable name in dataset	Number of missing values
Color of the movie	color	19
Name of the director	director_name	104
Number of external reviews	num_critic_for_review	50
Duration of the movie (in minutes)	duration	15
Facebook likes of the director's page	director_facebook_likes	104
Name of principal actor 1	actor_1_name	7
Name of principal actor 2	actor_2_name	13
Name of principal actor 3	actor_3_name	23
Facebook likes of actor 1's page	actor_1_facebook_likes	7
Facebook likes of actor 2's page	actor_2_facebook_likes	13
Facebook likes of actor 3's page	actor_3_facebook_likes	23
Box-office gross of the movie	gross	884
Budget of the movie	budget	492
Number of people featured on movie poster (on IMDb)	face_number_in_poster	13
Keywords describing the movie plot	plot_keywords	153
Number of users who gave a review (besides vote)	num_user_for_reviews	21
Language of the movie	language	12
Country of origin of the movie	country	5
Content rating of the movie	content_rating	303
Year when the movie was published	title_year	108
Aspect ratio of the movie	aspect_ratio	329
Genre classification of the movie	genres	0
Title of the movie	movie_title	0

Number of users who voted for the movie	num_voted_users	0
Total number of Facebook likes over all pages of the cast	cast_total_facebook_likes	0
Unique link to IMDb page of the movie	movie_imdb_link	0
Facebook likes of the movie's page	movie_facebook_likes	0
IMDb score of the movie	imdb_score	0

Table 7: List of all variables of the original IMDb 5000 dataset and their respective missing values

New Variables	
Variable	Name in the dataset
Number of movies directed (within the dataset) by the movie director	dir_app
Number of movies featuring (within the dataset) actor 1	actor_1_app
Number of movies featuring (within the dataset) actor 2	actor_2_app
Number of movies featuring (within the dataset) actor 3	actor_3_app
Binned Score (Type 1)	binned_score1
Binned Score (Type 2)	binned_score2
Action	Action
Adventure	Adventure
Biography	Biography
Comedy	Comedy
Documentary	Documentary
Drama	Drama
Familty	Familty
Fantasy	Fantasy
History	History
Horror	Horror
Music	Music
Mystery	Mystery
Romance	Romance

Sci-Fi	Sci-Fi
Sport	Sport
Thriller	Thriller
War	War
Western	Western

Table 8: List of newly created variables

Confusion Matrices

Logistic	truth			
Prediction	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1	0	0	0	0
Bin 2	17	113	46	0
Bin 3	11	139	531	25
Bin 4	0	0	4	14

Table 9: Confusion Matrix - Multinomial Logistic Regression

Random Forest	truth			
Prediction	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1	4	0	0	0
Bin 2	17	155	40	0
Bin 3	7	97	540	20
Bin 4	0	0	1	19

Table 10: Confusion Matrix - Random Forests

KNN	truth			
Prediction	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1	1	1	0	0
Bin 2	18	121	62	0
Bin 3	9	130	519	39
Bin 4	0	0	0	0

Table 11: Confusion Matrix - k-Nearest Neighbours

LDA	truth			
Prediction	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1	5	9	3	0
Bin 2	18	132	57	0
Bin 3	5	111	510	22
Bin 4	0	0	11	17

Table 12: Confusion Matrix - Linear Discriminant Analysis

SVM - Linear	truth			
Prediction	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1	0	0	2	0
Bin 2	24	164	56	2
Bin 3	5	93	505	22
Bin 4	0	0	3	24

Table 13: Confusion Matrix - Support Vector Machines: Linear Kernel

SVM - Polynomial	truth			
Prediction	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1	0	0	4	0
Bin 2	22	158	56	0
Bin 3	5	94	509	16
Bin 4	1	0	12	23

Table 14: Confusion Matrix - Support Vector Machines: Polynomial Kernel

SVM - Radial	truth			
Prediction	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1	2	6	1	0
Bin 2	25	174	64	0
Bin 3	2	77	495	20
Bin 4	0	0	6	28

Table 15: Confusion Matrix - Support Vector Machines: Radial Kernel

Boosting	truth			
Prediction	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1	6	5	1	0
Bin 2	17	169	33	0
Bin 3	5	78	544	17
Bin 4	0	0	3	22

Table 16: Confusion Matrix - Boosting

8.2 Figures

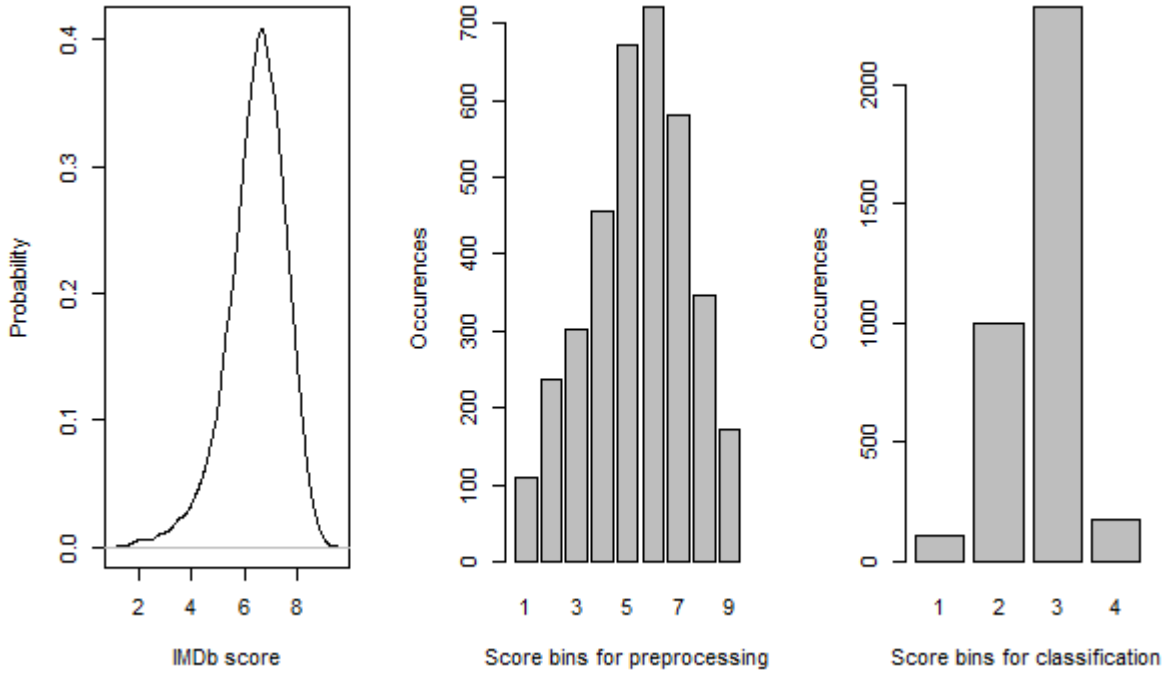


Figure 3: Density and distribution of binned scores

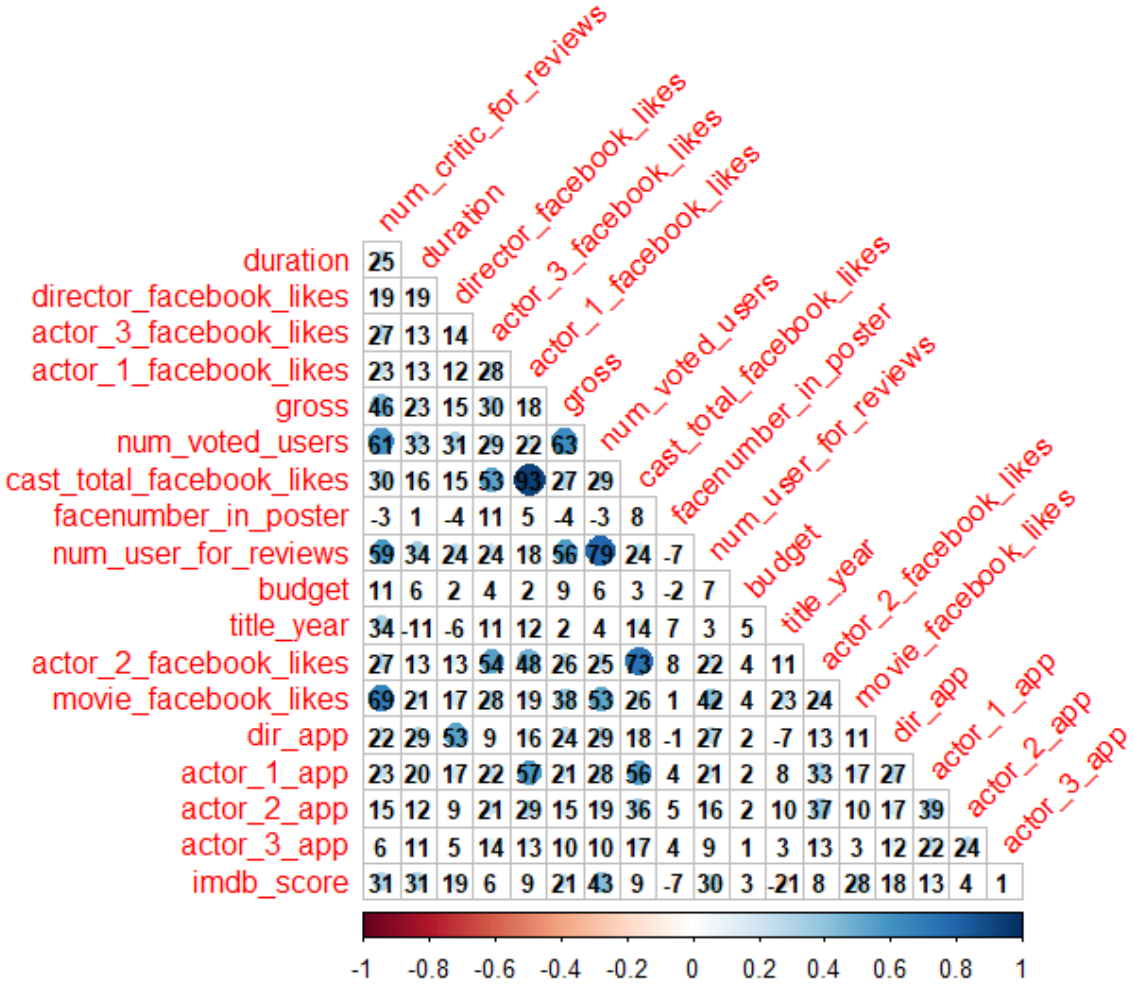
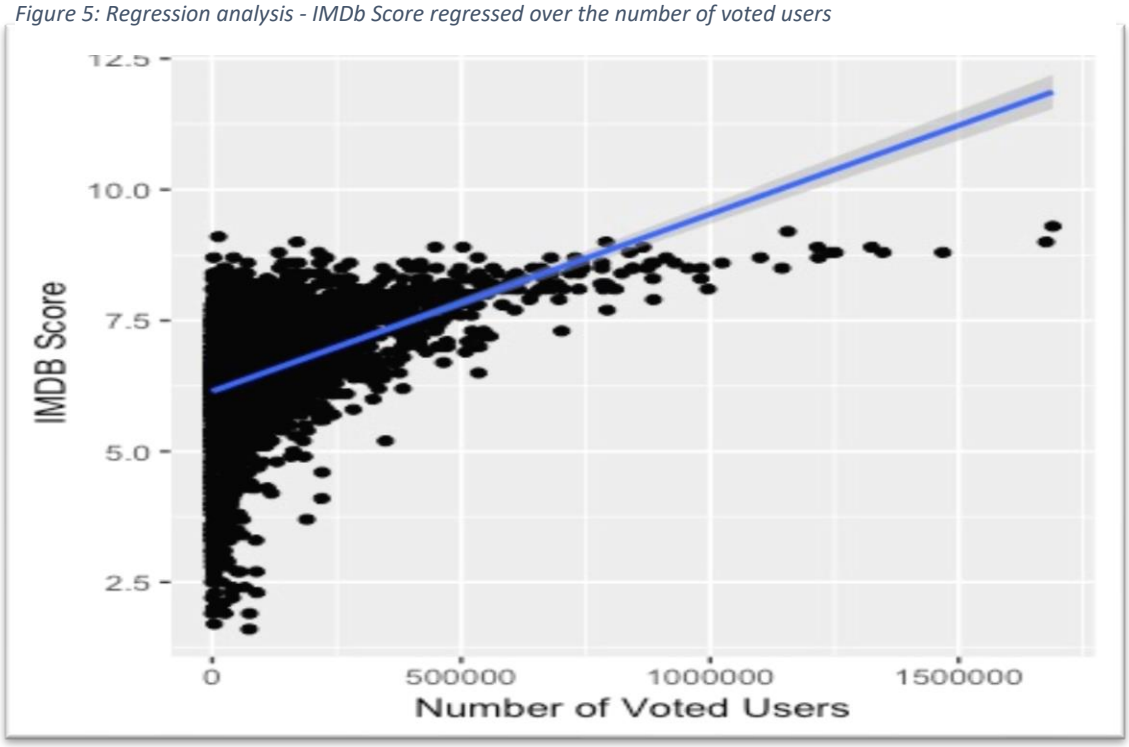


Figure 4: Correlation Plot between all the variables of IMDb 500 Dataset

Figure 5: Regression analysis - IMDb Score regressed over the number of voted users



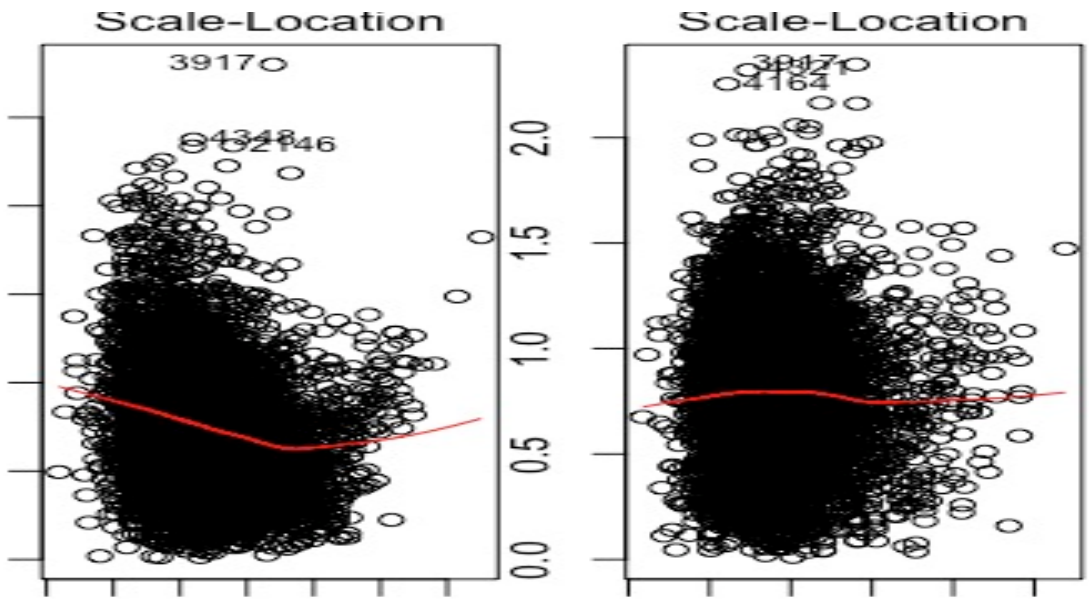


Figure 6: : Plots of √(Standardized Residuals) vs Fitted values before (left) and after (right) the Box Cox transform. Left: There is a slight indication of Heteroscedasticity as the residuals are not spread equally among the range of predictors. Right: The response has been transformed Using Box Cox and no there is no evidence of heteroscedasticity.

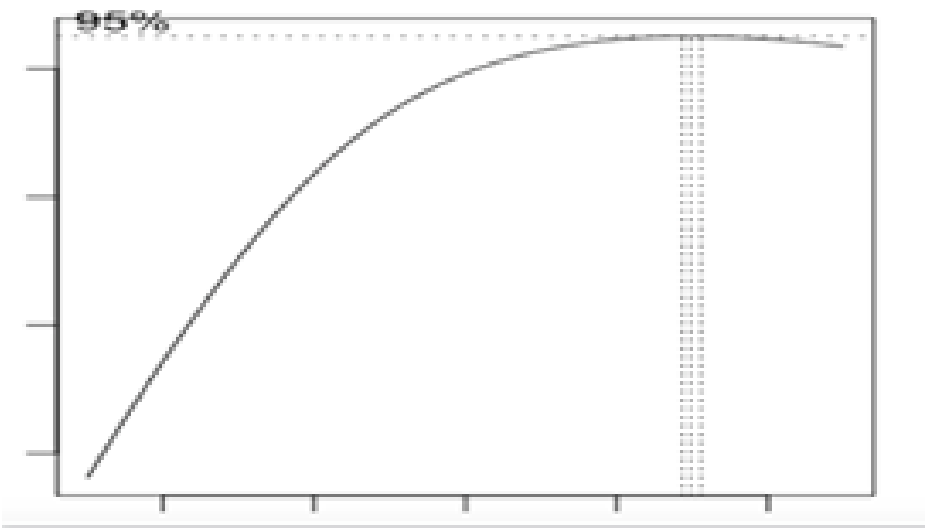


Figure 7: Caption: Box Cox "log-likelihood" vs lambda values

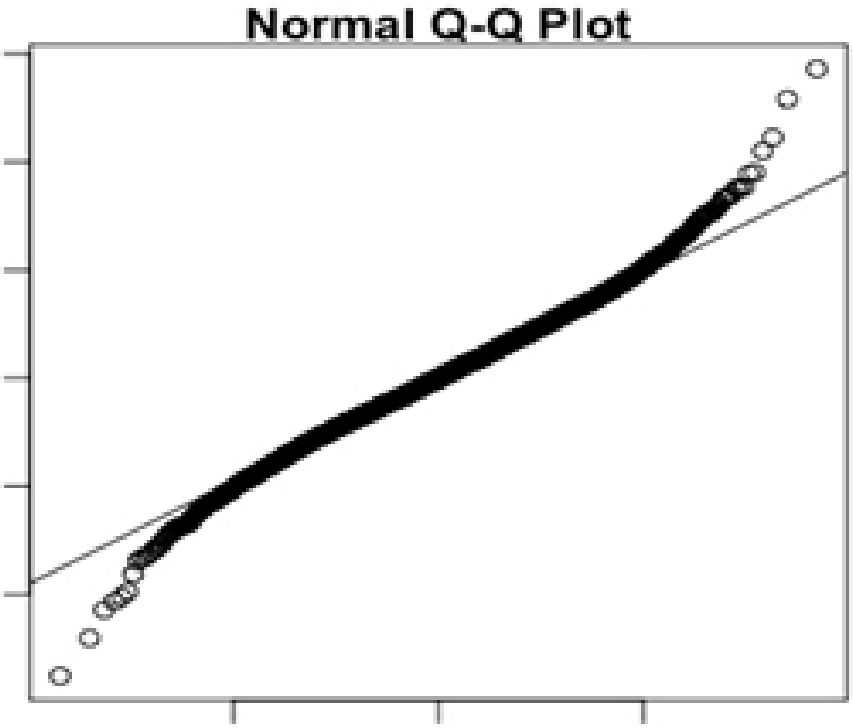


Figure 8: Normal probability plot for verifying normality