Predicting IMDB movie ratings

Sunday, November 5th 2017

Zachary Amar

Table of contents

Introduction

In this project, group *R we done yet* sought to predict the IMDB ratings of the following 9 movies which come out in early November: *Daddy's home 2*, *A Bad Moms Christmas*, *Thor Ragnarok*, *Murder On The Orient Express*, *Justice League: Unite the League*, *Wonder*, *Coco*, *Polaroid* and *The Star*.

In order to accurately predict the score of the following movies, *R we done yet* built a statistically significant predictive model based on scraped data from existing movies, with the goal of being capable of accurately predicting the IMDB score of the aforementioned movies.

Our objectives are to build a model with the highest possible *r-squared* - the higher the *r-squared* the better our model can explain can explain the variation in the IMDB score – while avoiding overfitting of the data, by having the lowest Mean Square of Errors (MSE) possible. Overfitting would prevent our model from accurately predicting the IMDB scores of the 9 movies previously mentioned.

We will first start by explaining how the dataset was cleaned and what errors were corrected. This will be followed by a data description, how the model was built and our model performance results. Last but not least, we will present our predictions regarding the nine upcoming movies.

Data Description

The movie dataset originally provided for analysis had 5043 observations. However, the dataset included unexpected values that did not prove to be conducive to analysing and predicting modern movies, and also contained many errors and other disorganizations that needed to be modified. We also created new variables to describe numerical aspects of categorical variables that would've been difficult to incorporate.

To overview the data, there are four broad categories: labels, the dependent variable, continuous predictors, and categorical predictors. Labels would include "movie_imdb_link" (the IMDB page for the movie) and "movie_title" (the title of the movie).

The dependent variable is the "imdb_score", which is the numeric score (from 0-10 inclusive) that a combination of members and critics of the IMDB community have given the movie. It is usually represented by stars on the IMDB site. All predictors are values that are found or linked to on the IMDB page for the movie, and can help to predict

the rating of the movie (as they can be what viewers consider before seeing the movie, and what factors influence viewer's perception of the movie).

Continuous predictors include "duration" (how long the movie is, in minutes), "director_facebook_likes", "actor_1_facebook_likes", "actor_2_facebook_likes", "actor_3_facebook_likes", "cast_total_facebook_likes", "movie_facebook_likes" (all of which represent the number of Facebook likes on the entities' Facebook pages), "facenumber_in_poster" (how many faces are shown in the movie poster), "year" (release year), and "budget" (how much the movie cost).

Categorical variables include "director_name" (the name of the movie director), "actor_1_name", "actor_2_name", "actor_3_name" (names of the actors), "genres" (genres associated with the movie), "plot_keywords" (key aspects of the movie), "language" (what language the movie is in), "country" (what country the movie was released in), "content_rating" (what audience is allowed to view the movie), and "aspect_ratio" (the proportional relationship between width and height that the movie was presented in). Many of the variables had very high max values, but many were very positively skewed (like budget, and all of the Facebook likes because Facebook has only been out so long). Many of the variables were also numerical but integer values, like facenumber_in_poster, which makes them harder to fit a model to.

The first stage of data cleaning that we performed on the data was related to literal cleaning of noticeable errors. All movie titles had the character "Ê", so we removed it. For country, we changed "New Line" and "Official Site" to "USA", and changed blank to USA, as all the movies that had those values for county were scraped erroneously and were USA.

The second stage of data cleaning we performed was to create several new variables. This focused on the variables "plot_keywords", "movie_title", and "genre". The plot keywords and genre variables were delimited to separate out the parts - "plot_keywords" has five keywords per cell in the original dataset, and "genre" has up to eight. We then chose to create a variable for genre that counted how many genres were defined for a movie, called "genre_count". We also created plot_keyword_length that defined the average character length of the plot keywords (or phrases), and created movie_title_length that defined the character length of the movie title. All variables that were based on character counts included spaces and grammar and punctuation as characters. For example, a movie with the genres "Horror" and "Crime", plot keywords "murder" and "suspense", and a title "Who Survives?" would have a genre count of two,

plot_keyword_length of seven, and a movie_title_length of 13. We thought that the complexity of the words associated with the movie might have a contributory effect on the rating of the movie - maybe ones with more complex keywords would have a different appeal versus movies with less complex ones. Lastly, we created a pseudo primary key from the variables title_year, movie_title, and duration. We attempted to create a variable that noted whether or not a movie was a sequel, but an accurate method was not forthcoming.

The third stage of data cleaning we performed was to remove movies that met certain conditions. We started off by removing all movies that had TV-related "content_rating", as they were not truly movies and were thus irrelevant to our analysis. That prompted us to look at the definition of a movie length, and by most major film entity standards feature films are movies 40 minutes or longer (reference rating system). While the average film runs between 70-210 minutes, we decided to leave the removal of higher duration movies to outlier testing. We also removed movies with less than 30 representations of any content ratings due to sample size issue, and remove movies with the content rating "Approved" because it was deemed to be too unrelated to the current ratings era. That finding prompted us to remove movies with a title_year from before the modern ratings era, which is 1990 onwards, according to the Motion Picture Association of America system (reference feature film). We decided those ratings would not contribute to our modern movie analysis because they would cause problems when trying to fit our model to their associated data and more modern movies. We also removed movies that had "Reality-TV" or "Game-Show" genres. Lastly, we removed duplicates based on the primary key we created.

The fourth stage of our data cleaning was to transform variables. We examined the variable language and found that more than 90% of movies were in the English language, so we converted language to a binary variable called langauge_code (0 being other languages and 1 being English). When we were testing our model later on we had to revisit the data and also input "No Genre" for the blank cells relating to the various genre columns ("genre_1", "genre_2", etc).

The fifth and last stage of our data cleaning was to group variables. As mentioned previously, when testing our model we had to revisit the data. We had a specific error relating to categorical variables, when we split the data to test the model levels were not represented in both the training and the testing samples. Because the sampling is decided at random, we chose to transform the genre columns and country column data. After

analyzing the representation of the different genres in the different columns and noting that the genres appeared alphabetically in each movie (and not by primary genre), we chose to replace genres that represented less than 3% of entries with "Other". For example, the column "genre_8" had values of "No Genre", "Romance" and "Thriller". The latter two values were each representative of less than 3% of movies, so were correspondingly replaced with "Other". We took the same approach with country - countries that represented less than 2% of overall countries were grouped together. This left us with the values "USA", "France", "Canada", "UK", "Germany", and "Other". Subsequently, the likelihood that a particular level wouldn't be represented in the testing and training data splits was decreased.

After cleaning and transforming our data we were left with 3565 movies to create a model with. We also re-ordered to columns to make our analysis easier, and put all numerical columns first, categorical columns second, and unused text columns last. The unused columns included "movie_imdb_link", "director_name", "actor_1_name", "actor_2_name", and "actor_3_name". Related, while performing our analysis in R, we ran the command na.omit(). This removed NA values that were hampering our analysis, which left us with 3490 movie observations.

Model Selection

Before we even started building the model, we started off by dealing with outliers. We went variable by variable and performed an outlier test. We then removed the rows that contained these outliers to obtain a clean data set. We also ran a total regression outlier and removed the lines that represented outliers to the overall model. (See Appendix 1 for the q-q plot)

The next step was to check for collinearity. We started off using a matrix of correlation (see Appendix 2). After having analysed it, we realised that the variables total actor like and total likes showed signs of collinearity consequently excluded them from our model.

After having removed the outliers and tested for collinearity, we ran a regression with all the variables with the objective of determining what factors were significant. After looking at the p-values that each coefficients gave us we decided to keep all the quantitative ones that had a p value smaller than 0.05. The quantitative values included: duration, director's Facebook likes, actor 1 Facebook likes, movie Facebook likes, face

number in poster, title year, movie title length and language code. However, we realised that it did not make sense to use the movie Facebook likes from movies that still had not yet come out. Even if by removing the movie Facebook likes variables our r-squared went down we chose not to use the variable, as it had an illogical effect on the IMDB movie rating, and decided to remove it from the predicting model.

We then tested for heteroscedasticity, to make sure that the values were actually correlated. After using a coeftest (see Appendix 3) we realised that title year was actually not very influential and removed it from the model.

The next step was to determine what type of effect each variable had on the total model. We started off by plotting all the variables (see Appendix 4) against the IMDB rating and observed the resulted graphs.

We first realised that the language code had only two options wo we decided that this should be a linear factor. We confirmed that hypothesis by performing a tukey test which gave a p-value of 0.203 and confirming our hypothesis.

We then realised that the values for director Facebook were very segmented therefore we decided to use spline regression. We set knots at 20%, 40%, 60% and 80% of the data as we believed that it was a good spread to represent the large amount of data skewed to the right.

The final part was deciding what polynomial degree to apply to the other factors. To do so we used an analysis of variance and looked at the p-value until it got over 0.05. The final results were that duration was a 10th degree polynomial, actor 1 Facebook likes and movie title length were fifth degree polynomials while face numbers in the poster and title year were second degree polynomials.

Finally, we added the categorical variables one by one and observed if the adjusted r-squared went up, as this value only goes up if the extra variable actually increases the predictive power of the model. We had to do this separately because we were not able to tell the effect of the many categories when testing the overall model. We therefore added: content rating, country and all the genres, from 1 through 8.

Results

Our final model included the numerical variables duration, director's Facebook likes, actor 1 Facebook likes, face number in poster, movie title length, and language code. It also included the categorical variables content rating, country and all of the

genre_n variables. For the summary statistics of the model and the specific variable manipulation within the model please refer to Appendix 6. Our residuals were reasonable, with the core range from the first quartile to the third varying from about -.5 to .5, but the min and max residuals were larger.

Some direct conclusions can be drawn for the variables. For content rating, all variables can affect the rating by up to 1 star more or less, and it seems that most ratings appear to improve the score of the movie except for PG-13. Unrated seems to be the rating that best improves the score of the movie, while PG-13 detracts from the potential score of the movie.

For country, all countries except Germany seem to produce better-rated movies than the base comparison USA, especially the United Kingdom. This could be because there's so many movies that come out from the US that it's a quantity-over-quality type problem. This conjecture is reinforced by the variable language code, which shows that the movie language being English decreases the rating by .75.

For all the genre variables, "Other" was used as the base comparison, and genres appeared across the genre variables in alphabetical order (and not by primary genre). Therefore, it's hard to conclude much directly from genre, but for each variable the movie may improve or not in score based on the genre that happens to be assigned to that variable. For example, genre_2 has a base comparison of Action as the genre, which means that movies with Comedy, Drama, Fantasy, or Horror in that genre slot will be rated better, and the others worse.

Unfortunately, the polynomial variables are hard to draw direct conclusions from, as their impact on the score can vary widely depending on the number of degrees and slope. However, they were chosen based on an ANOVA test to include a polynomial degree that adds to our model without losing significance (p-value < .05).

Our final model has a residual standard error of 0.912 on 3486 degrees of freedom, and an r-squared of 0.2931. It had an F-statistic of 18.53 on 78 and 3486 DF with a p-value of less than 2.2e-16, therefore the model is significant. However, this model does not have an overall strong predictive power. The r-squared value of 0.2931 indicates that only 29.31% of the movie's IMDB score can be explained by our model.

The model has a mean square error of around 2.44, which means that for any prediction the model makes the prediction will likely be over or under by 2.44 stars. Given that, our model is not very useful in predicting movie ratings, as there is a huge difference

between a predicted 6-star movie that got 8 stars and a predicted 8-star movie that got 6 stars.

Furthermore, if we add the fact that our model has such a low r-squared value, it is normal that our predictions aren't very precise. There are multiple factors that can be affecting our model. First, the nature of the prediction means that a good model has to be extremely precise, being able to predict results with a mean square error that is under 1. For us, our final model takes into account a lot of different variables, which are mostly analysed separately and then put together. We think that this is why the r-squared is smaller and MSE larger. Second, a model that is chosen more arbitrarily with less variables and more basis on current movies might do better than our complex model.

Predictions and Conclusions

Movie title	***Prediction***
Daddy's Home 2	5.45
Bad Moms Christmas	5.82
Thor-Ragnarok	6.34
Murder On The Orient Express	6.08
Wonder	6.71
Coco	7.25
Polaroid	6.19
The Star	6.34
Justice League	54652.06

The results shown in the above table are the IMDB scores R we done Yet, expects the upcoming nine movies to get upon their theatre release.

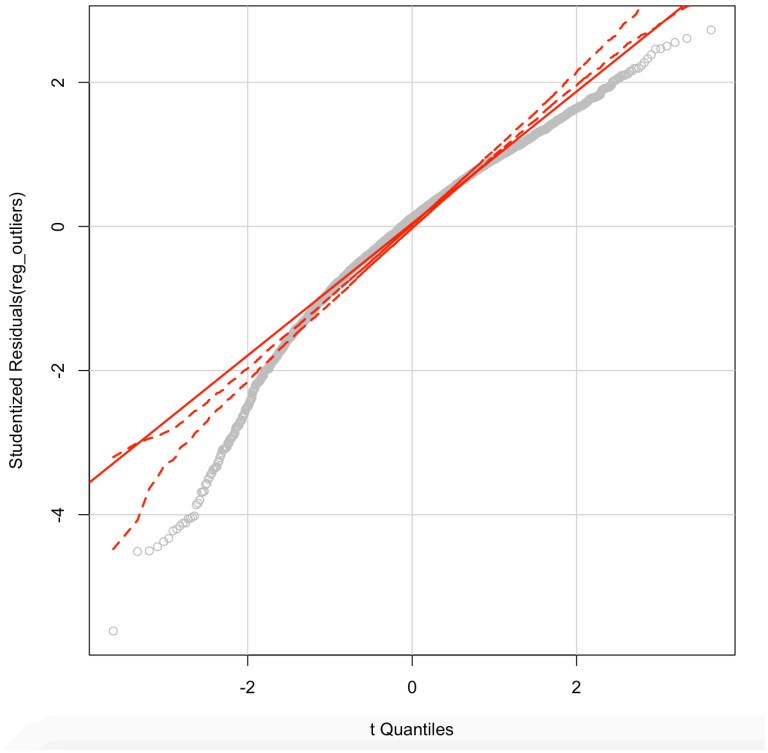
However, we do not believe that our model is predicting accurate scores. First of all, our model has an R-squared of .2931. This is not very high. We cannot explain most of the error that would come up. This could be due to a lack of data on variables (ie movie film studio, people past and current genre preferences).

Also, we have an MSE of 2.44. While it does not seem like a big MSE, 2.44 rating up or down in the rating out of 10 is pretty high and thus our prediction might not be the most accurate. This could signal that our model might suffer from slight over fitting, which can be explained with a 10th degree polynomial for duration.

Our main issue remains that our model was proven unsuccessful in predicting the IMDB score of Justice League. An IMDB score of 10+ makes no sense. When looking at the information about this movie compared to the others, we realise that it has a lot more director Facebook likes and actor 1 Facebook likes. In fact, it has no many director Facebook likes that it goes over the max, 23000, that our database records. The same thing happens for actor 1 Facebook likes. It is above the maximum number of likes. This leads to extrapolation and thus our model is unsuccessful. If we input the maximum values instead of the actual values, our values we actually get a rating of 6.86, a value which makes a lot more sense.

Appendices

Appendix 1: *Q-q Plot*

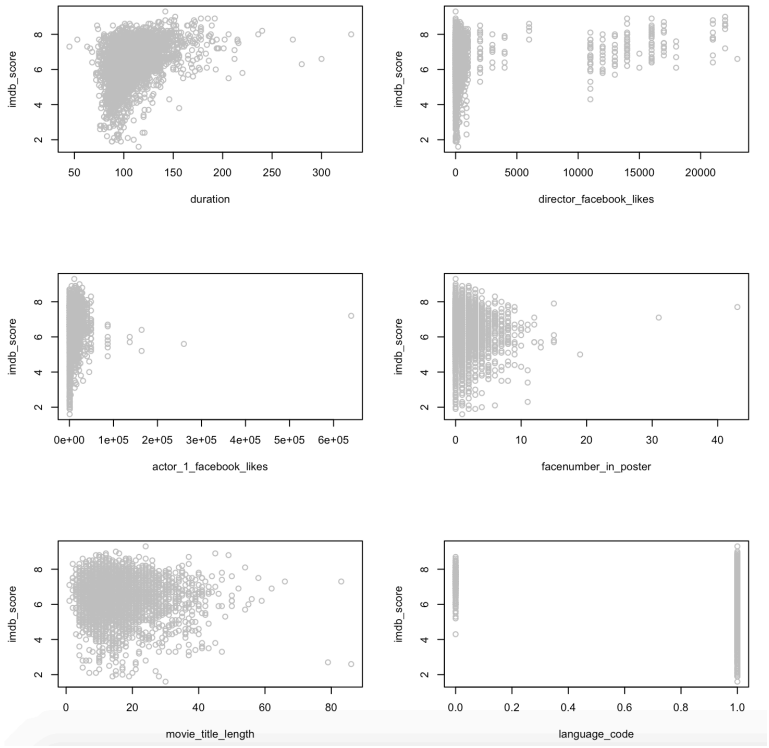


Appendix 2: *Matrix of Correlation*

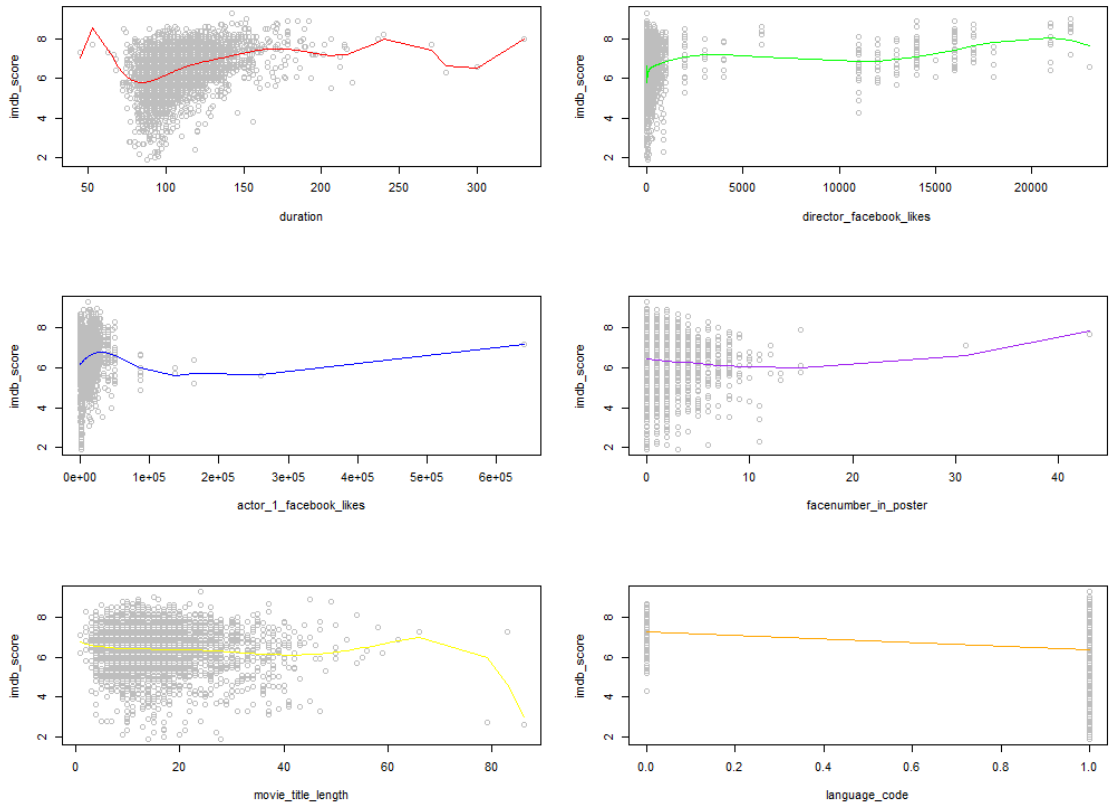
Appendix 3: *Values tested for heteroscedasticity*

	Dependent variable:
duration	0.014***
	(0.001)
director_facebook_likes	0.00003***
	(0.00000)
actor_1_facebook_likes	0.00000***
	(0.00000)
movie_facebook_likes	0.00001***
	(0.00000)
facenumber_in_poster	-0.027***
	(0.010)
title_year	-0.023***
	(0.003)
movie_title_length	-0.008***
	(0.002)
language_code	-0.917***
	(0.061)
Constant	52.005***
	(5.642)
Note:	*p<0.1; **p<0.05; ***p<0.01

Appendix 4: *Used independent variables plotted against IMDB rating*



Appendix 5: *Independent variables plotted against IMDB rating with model fit line*



Appendix 6: *Data cleaning references*

Feature Film - https://en.wikipedia.org/wiki/Feature_film
Rating System - https://en.wikipedia.org/wiki/Motion_Picture_Association_of_America_film_rating_system

R-Code

```
detach(Movie_Dataset)
attach(Movie_Dataset)
Movie_Dataset = na.omit(Movie_Dataset)

#install and require packages
install.packages("psych")
install.packages("car")
install.packages("olsrr")
install.packages("lmtest")
install.packages("plm")
install.packages("dplyr")
install.packages("data.table")
install.packages("caret")
install.packages("caTools")
install.packages("splines")
install.packages("stargazer")
require(stargazer)
require(splines)
require(caTools)
require(caret)
require(psych)
require(olsrr)
require(car)
require(lmtest)
require(plm)
require(dplyr)
require(data.table)
require(boot)

################
#Outlier Testing
################

#Individual Outlier Testing
reg_outliers1 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$duration)
outlierTest(reg_outliers1)
Movie_Dataset = Movie_Dataset[-c(2443, 2163), ]

reg_outliers2 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$director_facebook_likes)
outlierTest(reg_outliers2)
#Movie_Dataset = Movie_Dataset[-c(320), ]

reg_outliers3 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$actor_1_facebook_likes)
outlierTest(reg_outliers3)
Movie_Dataset = Movie_Dataset[-c(1140), ]

reg_outliers4 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$actor_2_facebook_likes)
outlierTest(reg_outliers4)
Movie_Dataset = Movie_Dataset[-c(114), ]

reg_outliers5 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$actor_3_facebook_likes)
outlierTest(reg_outliers5)
#Movie_Dataset = Movie_Dataset[-c(332), ]

reg_outliers6 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$total_actor_likes)
outlierTest(reg_outliers6)
#Movie_Dataset = Movie_Dataset[-c(1047), ]
```

```
reg_outliers7 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$movie_facebook_likes)
outlierTest(reg_outliers7)
#Movie_Dataset = Movie_Dataset[-c(707), ]

reg_outliers8 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$total_likes)
outlierTest(reg_outliers8)
#Movie_Dataset = Movie_Dataset[-c(878), ]

reg_outliers9 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$facenumber_in_poster)
outlierTest(reg_outliers9)
#Movie_Dataset = Movie_Dataset[-c(1656), ]

reg_outliers10 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$budget)
outlierTest(reg_outliers10)
#Movie_Dataset = Movie_Dataset[-c(445), ]

reg_outliers11 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$title_year)
outlierTest(reg_outliers11)
#Movie_Dataset = Movie_Dataset[-c(776), ]

reg_outliers12 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$aspect_ratio)
outlierTest(reg_outliers12)
Movie_Dataset = Movie_Dataset[-c(490), ]

reg_outliers13 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$movie_title_length)
outlierTest(reg_outliers13)
#Movie_Dataset = Movie_Dataset[-c(703), ]

reg_outliers14 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$country_code)
outlierTest(reg_outliers14)
#Movie_Dataset = Movie_Dataset[-c(142), ]

reg_outliers15 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$language_code)
outlierTest(reg_outliers15)
#Movie_Dataset = Movie_Dataset[-c(775), ]

reg_outliers16 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$plot_keyword_avg_length)
outlierTest(reg_outliers16)
#Movie_Dataset = Movie_Dataset[-c(826), ]

reg_outliers17 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$genre_count)
outlierTest(reg_outliers17)
#Movie_Dataset = Movie_Dataset[-c(2712), ]

reg_outliers18 = lm(Movie_Dataset$imdb_score ~ Movie_Dataset$content_rating_code)
outlierTest(reg_outliers18)
Movie_Dataset = Movie_Dataset[-c(1048), ]

#outlier regression model
reg_outliers = lm(imdb_score ~ duration + director_facebook_likes + actor_1_facebook_likes
            + actor_2_facebook_likes + actor_3_facebook_likes + total_actor_likes +
movie_facebook_likes
            + total_likes + facenumber_in_poster + budget + title_year + aspect_ratio +
movie_title_length
            + country_code + language_code + plot_keyword_avg_length + genre_count +
content_rating_code)
#outlier test
outlierTest(reg_outliers)
#outlier visual test
```

```
qqPlot(reg_outliers, col="grey")

#remove outlier rows
Movie_Dataset <- Movie_Dataset[-c(1658, 2720, 2611), ]

#Test for colinearity
quantvars=Movie_Dataset[,c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19)]
pairs.panels(quantvars)
  #From this we know not to use total_actor_likes and total likes

####################
#Model Construction
####################

#relevel categorical variables

genre_1 = as.factor(genre_1)
genre_2 = as.factor(genre_2)
genre_3 = as.factor(genre_3)
genre_4 = as.factor(genre_4)
genre_5 = as.factor(genre_5)
genre_6 = as.factor(genre_6)
genre_7 = as.factor(genre_7)
genre_8 = as.factor(genre_8)
country = as.factor(country)
content_rating = as.factor(content_rating)
genre_1 = relevel(genre_1, ref="Other")
genre_2 = relevel(genre_2, ref='Other')
genre_3 = relevel(genre_3, ref='Other')
genre_4 = relevel(genre_4, ref='Other')
genre_5 = relevel(genre_5, ref='Other')
genre_6 = relevel(genre_6, ref='Other')
genre_7 = relevel(genre_7, ref='Other')
genre_8 = relevel(genre_8, ref='Other')
country = relevel(country, ref='USA')
content_rating = relevel(content_rating, ref='PG')

#Multiple linear reg with all variables
all_variable_reg = lm(imdb_score ~ duration + director_facebook_likes + actor_1_facebook_likes
                  + actor_2_facebook_likes + actor_3_facebook_likes + movie_facebook_likes
                  + facenumber_in_poster + budget + title_year + aspect_ratio
                  + movie_title_length + language_code + plot_keyword_avg_length + budget
                  + genre_count + content_rating + content_rating_code + genre_1 + genre_2 + genre_3
                   + genre_4  + genre_5 + genre_6 + genre_7 + genre_8 + country)
summary(all_variable_reg, type = "html")

#From this we keep the strongly correlated (1-2-3 stars) - excpet movie facebook likes
correlated_variable_reg = lm(imdb_score ~ duration + director_facebook_likes + actor_1_facebook_likes
                       + facenumber_in_poster + title_year + movie_title_length
                       + language_code)
summary(correlated_variable_reg)

#Test for heteroskedasticity
ncvTest(correlated_variable_reg)
#since the p value is smaller than 0.05 then there is heteroskedasticity
#Correcting heteroskedasticity
coeftest(correlated_variable_reg, vcov = vcovHC(correlated_variable_reg, type = "HC1"))
  #From this we realise title year is not actually important

#Ajusting for polynomials
```

```
#plot all to deicde what method to use
par (mfrow=c(2,3))
plot(duration, imdb_score, col=c("grey"))
plot(director_facebook_likes, imdb_score, col=c("grey"))
plot(actor_1_facebook_likes, imdb_score,col=c("grey"))
plot(facenumber_in_poster, imdb_score,col=c("grey"))
plot(movie_title_length, imdb_score,col=c("grey"))
plot(language_code, imdb_score,col=c("grey"))
par (mfrow=c(1,1))

#Conclusions:

  #language code - leave as linear from binary -- confired with tukey test
  language = lm(imdb_score ~ language_code)
  residualPlots (language)
  par (mfrow=c(1,1))

  #director facebook likes -- use knot as segmentaed data
  quantile(director_facebook_likes, c(.20, .40, .60,.80))
  #either at 6,33,96,287

  #Actor 1 Facebooklikes
  a11 = lm(imdb_score ~ poly(actor_1_facebook_likes, 1))
  a12 = lm(imdb_score ~ poly(actor_1_facebook_likes, 2))
  a13 = lm(imdb_score ~ poly(actor_1_facebook_likes, 3))
  a14 = lm(imdb_score ~ poly(actor_1_facebook_likes, 4))
  a15 = lm(imdb_score ~ poly(actor_1_facebook_likes, 5))
  a16 = lm(imdb_score ~ poly(actor_1_facebook_likes, 6))
  anova(a11,a12,a13,a14,a15,a16) #a1 is 5th

    #movie title length
  mtl1 = lm(imdb_score ~ poly(movie_title_length,1))
  mtl2 = lm(imdb_score ~ poly(movie_title_length,2))
  mtl3 = lm(imdb_score ~ poly(movie_title_length,3))
  mtl4 = lm(imdb_score ~ poly(movie_title_length,4))
  mtl5 = lm(imdb_score ~ poly(movie_title_length,5))
  mtl6 = lm(imdb_score ~ poly(movie_title_length,6))
  anova(mtl1,mtl2,mtl3,mtl4,mtl5,mtl6) ##mtl is 5th

    #Duration
  dur1 = lm(imdb_score ~ poly(duration,1))
  dur2 = lm(imdb_score ~ poly(duration,2))
  dur3 = lm(imdb_score ~ poly(duration,3))
  dur4 = lm(imdb_score ~ poly(duration,4))
  dur5 = lm(imdb_score ~ poly(duration,5))
  dur10 = lm(imdb_score ~ poly(duration,10))
  dur11 = lm(imdb_score ~ poly(duration,11))
  anova(dur1,dur2,dur3,dur4,dur5,dur10, dur11) #use 10th degree

    #Facenumber in poster
  fnp1 = lm(imdb_score ~ poly(facenumber_in_poster,1))
  fnp2 = lm(imdb_score ~ poly(facenumber_in_poster,2))
  fnp3 = lm(imdb_score ~ poly(facenumber_in_poster,3))
  fnp4 = lm(imdb_score ~ poly(facenumber_in_poster,4))
  fnp5 = lm(imdb_score ~ poly(facenumber_in_poster,5))
  fnp6 = lm(imdb_score ~ poly(facenumber_in_poster,6))
  anova(fnp1,fnp2,fnp3,fnp4,fnp5,fnp6) # use 2nd degree

#Graphs with the fitted line
```

```
 par (mfrow=c(2,3))
 plot(duration, imdb_score, col=c("grey"))
 lines(sort(Movie_Dataset$duration), predict(dur10)[order(Movie_Dataset$duration)], col="red")
 plot(director_facebook_likes, imdb_score, col=c("grey"))
 lines(sort(Movie_Dataset$director_facebook_likes),
predict(dfb)[order(Movie_Dataset$director_facebook_likes)], col="green")
 plot(actor_1_facebook_likes, imdb_score,col=c("grey"))
 lines(sort(Movie_Dataset$actor_1_facebook_likes),
predict(a15)[order(Movie_Dataset$actor_1_facebook_likes)], col="blue")
 plot(facenumber_in_poster, imdb_score,col=c("grey"))
 lines(sort(Movie_Dataset$facenumber_in_poster),
predict(fnp2)[order(Movie_Dataset$facenumber_in_poster)], col="purple")
 plot(movie_title_length, imdb_score,col=c("grey"))
 lines(sort(Movie_Dataset$movie_title_length),
predict(mtl5)[order(Movie_Dataset$movie_title_length)], col="yellow")
 plot(language_code, imdb_score,col=c("grey"))
 lines(sort(Movie_Dataset$language_code), predict(language)[order(Movie_Dataset$language_code)],
col="orange")
 par (mfrow=c(1,1))

#Final Model
final_variable_reg = lm(imdb_score ~ poly(duration, 10) + bs(director_facebook_likes,
knots=c(6,33,96,287), degree=4)
                        + poly(actor_1_facebook_likes, 5) + poly(facenumber_in_poster,2)
                        + poly(movie_title_length, 5) + language_code + content_rating + country+ genre_1 +
genre_2 + genre_3 + genre_4
                        + genre_5 + genre_6 + genre_7 + genre_8)
summary(final_variable_reg)

########
#TESTING
########

#K-Fold test

Kfitlin=glm(imdb_score ~ poly(duration, 10) + bs(director_facebook_likes, knots=c(6,33,96,287),
degree=4)
        + poly(actor_1_facebook_likes, 5) + poly(facenumber_in_poster,2)
        + poly(movie_title_length, 5) + language_code + content_rating + country+ genre_1 + genre_2 +
genre_3 + genre_4
        + genre_5 + genre_6 + genre_7 + genre_8)




cv.error_kf=rep(0,500)
for (i in 1:500)
{
 cv.error_kf[i]=cv.glm(Movie_Dataset, Kfitlin, K=100)$delta[1]
}
mean(cv.error_kf)

############
#Predictions
############

value_Daddyshome2=data.frame(duration=100, director_facebook_likes=0,
actor_1_facebook_likes=105518, facenumber_in_poster=10, movie_title_length=14,language_code=1,
content_rating="PG-13", country="USA", genre_1="Comedy", genre_2="No Genre", genre_3="No
```

```
Genre", genre_4="No Genre", genre_5="No Genre", genre_6="No Genre", genre_7="No Genre",
genre_8="No Genre")
predict(final_variable_reg, value_Daddyshome2, type="response")

value_justiceleague=data.frame(duration=121, director_facebook_likes=183648,
actor_1_facebook_likes=1527000, facenumber_in_poster=5, movie_title_length=14,language_code=1,
content_rating="PG-13", country="USA", genre_1="Action", genre_2="Adventure", genre_3="Fantasy",
genre_4="Other", genre_5="No Genre", genre_6="No Genre", genre_7="No Genre", genre_8="No
Genre")
predict(final_variable_reg, value_justiceleague, type="response")

value_badmomsxmas=data.frame(duration=104, director_facebook_likes=0, actor_1_facebook_likes=0,
facenumber_in_poster=6, movie_title_length=20,language_code=1, content_rating="PG-13",
country="USA", genre_1="Action", genre_2="Adventure", genre_3="Comedy", genre_4="No Genre",
genre_5="No Genre", genre_6="No Genre", genre_7="No Genre", genre_8="No Genre")
predict(final_variable_reg, value_badmomsxmas, type="response")

value_Thor=data.frame(duration=130, director_facebook_likes=0, actor_1_facebook_likes=0,
facenumber_in_poster=9, movie_title_length=14,language_code=1, content_rating="PG",
country="USA", genre_1="Action", genre_2="Adventure", genre_3="Comedy", genre_4="Other",
genre_5="No Genre", genre_6="No Genre", genre_7="No Genre", genre_8="No Genre")
predict(final_variable_reg, value_Thor, type="response")

value_MotOE=data.frame(duration=114, director_facebook_likes=0, actor_1_facebook_likes=0,
facenumber_in_poster=9, movie_title_length=28,language_code=1, content_rating="PG",
country="USA", genre_1="Crime", genre_2="Drama", genre_3="Mystery", genre_4="No Genre",
genre_5="No Genre", genre_6="No Genre", genre_7="No Genre", genre_8="No Genre")
predict(final_variable_reg, value_MotOE, type="response")

value_Wonder=data.frame(duration=113, director_facebook_likes=0, actor_1_facebook_likes=0,
facenumber_in_poster=1, movie_title_length=6,language_code=1, content_rating="PG",
country="USA", genre_1="Other", genre_2="No Genre", genre_3="No Genre", genre_4="No Genre",
genre_5="No Genre", genre_6="No Genre", genre_7="No Genre", genre_8="No Genre")
predict(final_variable_reg, value_Wonder, type="response")

value_Coco=data.frame(duration=113, director_facebook_likes=0, actor_1_facebook_likes=23000,
facenumber_in_poster=5, movie_title_length=4,language_code=1, content_rating="PG",
country="USA", genre_1="Other", genre_2="Adventure", genre_3="Comedy", genre_4="Family",
genre_5="Other", genre_6="Other", genre_7="Other", genre_8="No Genre")
predict(final_variable_reg, value_Coco, type="response")

value_Polaroid=data.frame(duration=88, director_facebook_likes=0, actor_1_facebook_likes=41030,
facenumber_in_poster=0, movie_title_length=8,language_code=1, content_rating="PG-13",
country="Canada", genre_1="Horror", genre_2="No Genre", genre_3="No Genre", genre_4="No
Genre", genre_5="No Genre", genre_6="No Genre", genre_7="No Genre", genre_8="No Genre")
predict(final_variable_reg, value_Polaroid, type="response")

value_TheStar=data.frame(duration=86, director_facebook_likes=675,
actor_1_facebook_likes=0,facenumber_in_poster=0, movie_title_length=8,language_code=1,
content_rating="G", country="USA", genre_1="Other", genre_2="Adventure", genre_3="Comedy",
genre_4="Family", genre_5="Other", genre_6="No Genre", genre_7="No Genre", genre_8="No Genre")
predict(final_variable_reg, value_TheStar, type="response")
```