# Predicting IMDB Scores using Regression

Abdel Elsayed - 260953424
Marc ElGarrayac - 260944754
Alia Abdelaziz - 261009429
Chris Aclimandos - 261003812
Ranya Ammor - 261029021

December 4, 2023

**Abstract**

IMDb is a platform often used by users which they can obtain relevant information about specific movies and articles. The database provided illustrates an insight on the popularity of the movie to professionals and the public. It's an interactive platform, allowing users to rate movies, as well as leave comments on message boards.

Furthermore, throughout this paper we gathered data from IMDb to build an algorithm which attempts to recognize historical patterns in the movie industry and predict the IMDb score of upcoming movies.

## 1 Introduction

In this IMDB prediction challenge, we're on a journey, guided by our statistical knowledge, to predict IMDb ratings for a dozen upcoming blockbusters. Our mission is all about blending data analysis and predictive modeling to create a model that delivers the most accurate prediction. We're striving for the lowest Mean Squared Error (MSE) and the highest R-squared value, which will act as our compass throughout this adventure, driving us to fine-tune our model until it's a spot-on predictor of public opinion.

To start this journey, we dug deep into the training dataset, examining the variables. We wanted to figure out which factors have a significant impact on movie ratings and whether they should play a starring role in our model. Our first hunch led us to focus on production characteristics like the director, distributor, cinematographer, and production company. These aspects are what we often consider when guessing how a movie might perform. A talented director and cinematographer, along with big distributors and production companies, often spell success. We transformed these variables into dummy variables, giving us more control. To gain a clearer picture, we turned to boxplots and histograms, creating visual snapshots of the variable distributions. This helped us identify the key players in our model and potential red flags, like non-linear relationships, heteroskedasticity, outliers, and collinearity, which we wanted to avoid to keep our model unbiased.

Consequently, we individually scrutinized each variable and made some educated guesses, we tested for non-linearity. Moving on, we attempted trying out different polynomial degrees to see which one fit best. Finally, then we had enough to begin testing our models, running regressions with chosen variables and optimum polynomial degrees. In this report we will analyze these steps into more detail, showcasing our model and final predictive results.

## 2 Data description

Starting our analysis, we initially focused on the distribution of our dependent variable, '*imdb_score*.' To accomplish this, we constructed a density plot (refer to *appendix 1.1*), which offers a visual estimation of our variable. This visualization enabled us to pinpoint IMDb scores that are more likely to be prevalent in our dataset. This insight holds particular significance as it empowers us to establish a threshold IMDb score that serves as a benchmark for defining what constitutes a "good" rating.

Table 1: Data Dictionary

| | |
|---|---|
| **Dependent Variable** | IMDb score (Y) |
| **Independent Variables** | *identifiers*, *film characteristics*, *cast characteristics*, *production characteristics* |

## 2.1 Feature Engineering

This is the process whereby we created new features by using available features and intuitive knowledge to help improve the performance of our model:

1. *Good Director*: a director is considered "good" if their average IMDb score is superior to that of 50% of the dataset.
2. *Good Cinematographer*: a cinematographer is deemed "good" if their average score surpasses that of 50% of the dataset.

   Furthermore, in order to complete our analysis of the production characteristics, we created dummy variables for the independent variables "distributor" and "production company". For the following, we used the 90th percentile as a threshold.
3. *Big Production Company*: production companies with larger number of movie titles than 90% others in the dataset
4. *Big Distributors*: distributors of those with a larger number of movie titles than 90% others in the dataset.

## 2.2 Preprocessing

Furthermore, to continue analyzing the independent variables, we made the decision to exclude movie identifiers: "movie_title," "movie_id," and "imdb_link." These identifiers, while essential for uniquely identifying each movie, do not play a significant role in determining a movie's score. Therefore, we opted not to include them in our analysis. Instead, we focused on categorizing the remaining independent variables, classifying them into two categories:

Table 2: Categorical and Numerical Variables for Regression Model

| **Categorical Variables** | **Numerical Variables** |
|---|---|
| maturity_rating | movie_budget |
| all_movie_genres | release_day |
| language | release_year |
| country | duration |
| release_month | aspect_ratio |
| actor1 | nb_news_articles |
| actor2 | nb_faces |
| actor3 | movie_meter_IMDBpro |
| plot_keywords | actor1_star_meter |
| colour_film | actor2_star_meter |
| | actor3_star_meter |

### 2.2.1 Numerical Variables

For the numerical variables, we ran summary statistics tables, and visualized box plots and histograms. While this analysis does not directly help us identify the most significant variables, it provides us with some hints and insights. For example, expernimenting with the movie_budget variable, the highest budget for a movie in our dataset is 55 000 000$ with 75% of the movies having a budget lower than 30 000 000$. This indicates that a significant portion of the movies in the dataset are produced with

lower budgets. In fact, the most common budget among the movies is around 20 973 774$. This could be seen in the box plot (*appendix 1.2*) and the histogram (*appendix 1.3*). In the box plot, we can identify a few outliers representing movies with high budgets compared to the majority. The histogram further emphasizes the budget's distribution with a right-skewed pattern. This reiterates that the majority of movies tend to have lower budgets with some exceptions. Understanding the distribution and characteristics of the independent variables like movie_budget, guides us to creating an effective model.

#### 2.2.2 Visualizing Relationships between Variables

After conducting individual analysis on our variables dataset, we decided to run linear regression to predict the dependent variable "imdb_score" based on the numerical independent variables. Analyzing the summary tables for each regression model yielded essential information including coefficients, R-squared values and p-values.

After running all the regressions, one variable stood out with the highest $R^2$ value of 16.86%: "duration". This was indeed the highest individual $\hat{R2}$ obtained from all the variables against imdb_score. Conversely, some variables displayed low $R^2$ values and were deemed statistically insignificant when predicting movie scores. These included ($\hat{R2}$):

- actor3_star_meter: 0.00001657
- actor 2_star_meter: 0.001465
- actor1_star_meter: 0.0008368

These low R-squared values suggest that the star meter for the first three actors are not individually significant factors when attempting to predict IMDb scores. They can be improved if part of a multiple regression model; however, their extremely low scores indicates how insignificant they are.

In each of these regression we then corrected for heteroskedasticity, if present. This is to ensure that our regression results were not biased when analyzing them and accurately reflected the relationship between the variables. Further, both visual and numerical methods were used. Visually, we examined residual plots to detect any funnel-shaped patterns. Numerically, we ran an ncvTest to statistically test for the presence of heteroskedasticity. In our analysis, we identified the following variables with evidence of heteroskedasticity: *movie_budget, duration, nb_news_articles, movie_meter_IMDBpro and actor3_star_meter*. This would ensure the enhancement of the reliability of our model estimates.

## 3 Feature Selection

To select our optimal model, we went through a few steps to ensure we would get the best one.

### 3.1 Collinearity

First, we tested for correlation between the variables to avoid collinearity problems. As per *appendix 2.1*, no pair of variables exhibited correlation above 0.5, indicating we could ignore this issue.

### 3.2 Individual For Loops for each Variable against IMDb score

After having looked at the relationships between variables, we ran a for loop through each of the continuous variables to find the poly degree with the lowest MSE. The for loop iterated between poly degrees 1:5 against each individual continuous variable against imdb_score in a regression model. Then, the model returned the optimum poly degree that returns the lowest MSE. Further, the loop limited overfitting by finding the best polynomial up to the 5th degree at most. A snippet of the code used can be found in *appendix 2.2*.

Henceforth, having found the lowest MSE model, we chose to keep the variables whose models had the highest predictive value, looking at the $R^2$ of each of those optimal polynomial models. The polynomial degrees that were used are summarized in *Table 3*, below.

Table 3: Polynomial Degrees for Variables

| **Variable** | **Polynomial Degree** |
|---|---|
| release_year | 3 |
| duration | 5 |
| movie_budget | 1 |
| nb_news_articles | 1 |
| nb_faces | 1 |
| movie_meter_IMDBpro | 1 |

### 3.3 Dummy Variables

There were also some interesting dummy variables, however, we could not simply run loops, given their nature. For those, we ran individual linear regressions. Here too, we kept the ones with the highest predictive value. These included the movie genre, whether or not the director was a good one, and whether or not the cinematographer was a good one, as previously defined. We found that these variables had the greatest r-squared values as well, indicating they are the best predictors for IMDB scores.

From there, we had a preliminary model. We then chose to test removing the predictor variables with the highest p-values and compared the results, keeping the model with the best result. Even so, the model was not optimum, and human error could have hindered the performance of the model. Hence, before reaching the conclusion of variables to include in the model a *backward elimination process* took place.

### 3.4 Backward Elimination

Backward Elimination entails an iterative procedure that commences with all potential variables. During each iteration, it removes the variable that, when eliminated, results in the least statistically significant degradation in the model. In simpler terms, the variables with the highest p-values, usually ¡ 0.05, are removed first and so on. Therefore, when the process removes a variable first it shows that it is insignificant towards the model. This process is repeated until there are no more variables that can be removed without causing a statistically significant decrease in fitness.

So, to further support the selection of our variables, we ran a backward elimination for 15 variables to use in our final model. This would in turn lead to the model with the lowest MSE, and with the best predictive power given it's constraints. 15 variables were chosen, and not more, so that the model does not over-fit. Also, we did not choose ¿15 so that the model does not suffer from under-fitting.

### 3.5 Cross Validation

The aim of this is to obtain the model with the best out-of-sample performance; in other words, predictive power. Therefore, we ran another loop to get the best polynomial degree for each variable. We wanted to determine if we should use:

1. a linear spline (*degree = 1*).
2. a quadratic spline (*degree = 2*); or
3. a cubic spline (*degree = 3*); or
4. a quartic spline (*degree = 4*); or
5. a quintic spline (*degree = 5*).

In other words, we wanted to find the optimal combination of (a,b,c,d,e,f). With that, in the same for loop, we ran a K-fold validation set (with K=5). Consequently, the loop now runs a 5-fold test for each combination of (a,b,c,d,e,f), and then determine which combination gives you the lowest MSE. In the end, this only confirmed the results we obtained through our original individual for loops, mentioned earlier.

# 4 Results

The final linear regression model for IMDb score used can be expressed as follows:

$$\begin{aligned}\text{IMDb\_score} =& \beta_0 + \beta_1 \cdot poly(release\_year, 1) \\ &+ \beta_2 \cdot poly(release\_year, 2) + \beta_3 \cdot poly(release\_year, 3) \\ &+ \beta_4 \cdot poly(duration, 1) + \beta_5 \cdot poly(duration, 2) \\ &+ \beta_6 \cdot poly(duration, 3) + \beta_7 \cdot poly(duration, 4) \\ &+ \beta_8 \cdot poly(duration, 5) + \beta_9 \cdot \text{movie\_budget} \\ &+ \beta_{10} \cdot \text{nb\_news\_articles} + \beta_{11} \cdot \text{nb\_faces} \\ &+ \beta_{12} \cdot \text{movie\_meter\_IMDBpro} + \beta_{13} \cdot \text{action} \\ &+ \beta_{14} \cdot \text{western} + \beta_{15} \cdot \text{sport} + \beta_{16} \cdot \text{drama} \\ &+ \beta_{17} \cdot \text{animation} + \beta_{18} \cdot \text{rated\_R} \\ &+ \beta_{19} \cdot \text{good\_cinematographer} + \beta_{20} \cdot \text{good\_director} \end{aligned} \tag{1}$$

Where:

$\beta_0$ is the intercept coefficient

$\beta_1, \beta_2, \beta_3$ are coefficients for the polynomial terms of release year (1st, 2nd, and 3rd degree)

$\beta_4, \beta_5, \beta_6, \beta_7, \beta_8$ are coefficients for the polynomial terms of duration (1st to 5th degree)

$\beta_9$ is the coefficient for movie_budget

$\beta_{10}$ is the coefficient for nb_news_articles

$\beta_{11}$ is the coefficient for nb_faces

$\beta_{12}$ is the coefficient for movie_meter_IMDBpro

$\beta_{13}$ is the coefficient for action

$\beta_{14}$ is the coefficient for western

$\beta_{15}$ is the coefficient for sport

$\beta_{16}$ is the coefficient for drama

$\beta_{17}$ is the coefficient for animation

$\beta_{18}$ is the coefficient for rated_R

$\beta_{19}$ is the coefficient for good_cinematographer

$\beta_{20}$ is the coefficient for good_director

The results shown in *appendix 3.1* illustrate the performance of the regression algorithm that we obtained after running a backward elimination process, leaving the 15 most significant variables and a for loop to acquire the optimum degree for variables using polynomial regression while maintaining the lowest MSE. This was the best performing algorithm, while spline regression fell shortly behind. While the former algorithm seems to be similarly performing in terms of correlation and $R^2$, the error metric (MSE) is much higher in the Spline compared to Polynomial and Linear regression, as seen in *table 4*. Overall, the linear model performs exceptionally well. The $R^2$ was established at 57% and an MSE of 0.56. Hence, the model explains 57% variation in the imbd_score by using the chosen variables. Leaving us with the best out-of-sample performance model we could develop.

At the confidence level 95%, all coefficients, except 1 seem to be statistically significant as showcased in *appendix 3.1* table above by the '**' and '***'. The variable that is not statistically significant at the confidence level of 95% is 'nb_faces'. The justification for having this variable in the model is, it seemed significant when running the backward elimination process as well as when running individual linear regressions against imdb_score. Also, looking at the variables where poly regression was used, with degree = x, some of the polynomial levels are not significant. However, tests were run previously to indicate the optimum degree to be used; therefore, that can be left as is. Furthermore, of all the 15 variables used in the model, release_year, duration, movie_budget, nb_news_articles, action, drama, animation, rated_R, good_cinematographer, and good_director were found to be the most significant features.

Table 4: Movie Names and Predicted IMDb Scores

| **Movie Title** | **Predicted IMDb Scores** |
|---|---|
| Dream Scenario | 5.465 902 |
| Napoleon | 6.479 738 |
| The Holdovers | 7.383 500 |
| The Dirty South | 5.696 922 |
| The Hunger Games: The Ballad of Songbirds and Snakes | 6.434 089 |
| Next Goal Wins | 5.640 755 |
| Trolls Band Together | 5.556 867 |
| Wish | 5.095 692 |
| Pencils vs Pixels | 5.647 189 |
| Thanksgiving | 5.446 884 |
| The Marvels | 4.673 495 |
| Leo | 5.971 322 |

After running the model against the test data set that was given, the results can be observed in *Table 4*. The scores are relatively low of each of the 12 movies releasing soon. The highest being 'The Holdovers' with an $imdb_score$ prediction of 7.4 and the lowest being 'The Marvels' with an $imdb_score$ prediction of 4.7. It is important to note that a possible reason for predictions being relatively low is that the majority of movies in the test data set do not have directors and cinematographers that are in the train data set, which leads to worse predictions. 'The Holdovers' have both a good director and good cinematographer explaining the high score; on the contrary, other movies either have both missing or one missing which affects the predicted score, as with 'The Marvels'.

# 5 Conclusion

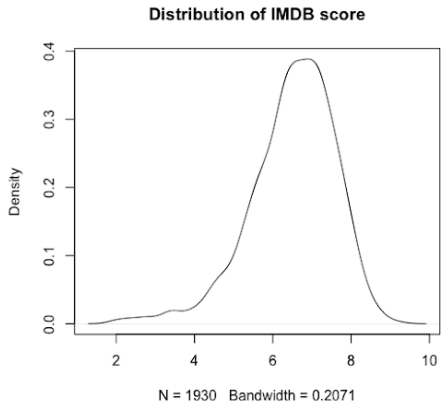
To conclude, the aim of this paper was to use regression techniques to predict imdb scores for upcoming movies using an imdb dataset provided. The final model presented a fair $R^2$ value of 57% and a low MSE of 0.56. Nevertheless, the predictor did demonstrate a slight negative bias with the test data and tended to assign ratings that seem to be lower than the actual ratings. However, this was due to a mismatch between datasets.

Overall, the method presented in this paper illustrates that imdb scores can be predicted using the model, with a good out-of-sample performance. Although some would argue the accuracy of the results will not be 100%; but, 57% of imdb score variability can be explained by the model. A way to attempt to improve the results of the model would be exploring different regressions such as Random Forest, LASSO, Ridge etc. that can aid in minimizing the MSE and increasing the $R^2$.

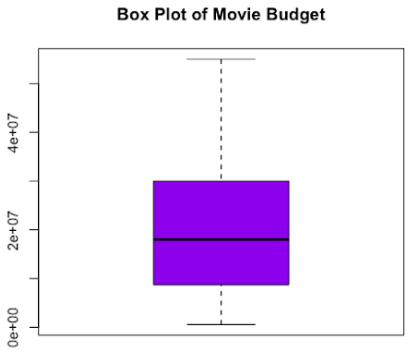
# 1 Appendix 1: Data description

This section includes all tables mentioned in *data description*

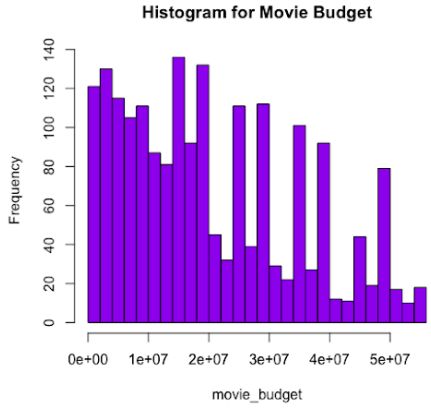
## 1.1 Appendix 1.1: Distribution of IMDb score




## 1.2 Appendix 1.2: Boxplot of Movie Budget



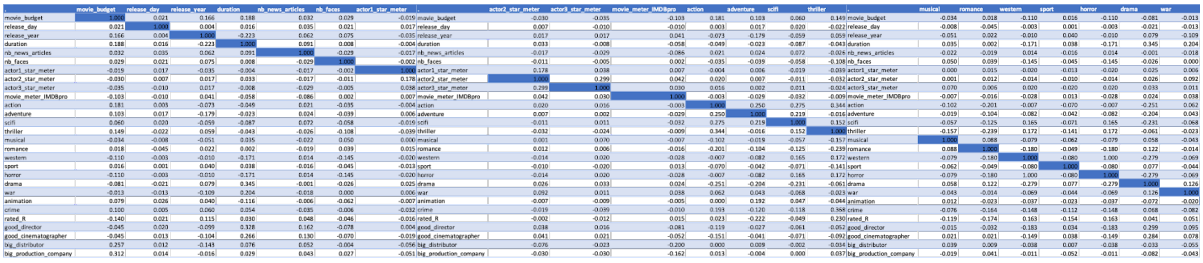

## 1.3 Appendix 1.2: Histogram for Movie Budget



# 2 Appendix 2: Feature selection

This section includes all tables mentioned in *feature selection*

## 2.1 Appendix 2.1: Correlation Matrix



## 2.2 Appendix 2.2: Code Snippet

```
best_degree1 = 1
best_mse1 = NULL
best_model1 = NULL

for (degree in 1:5) {
  model1 = glm(imdb_score ~ poly(movie_budget, degree, raw = TRUE))
  mse1 = cv.glm(data = IMDB_data, glmfit = model1, K = 5)$delta[1]

  if (is.null(best_mse1) || mse1 < best_mse1) {
    best_degree1 = degree
    best_mse1 = mse1
    best_model1 = model1
  }
}

cat("Best Degree:", best_degree1, "\n")
cat("Best MSE:", best_mse1, "\n")
summary(best_model1)
```

# 3 Appendix 3: Results

This section includes all tables mentioned in *results*

## 3.1 Appendix 3.1: Linear Regression Model Summary

Table 5: Linear Regression Model Summary

**Table 5: Linear Regression Model Summary**

| | *Dependent variable:* |
|---|---|
| | imdb_score |
| releaseyear[1] | −5.899*** |
| | (0.821) |

**Table 5: Linear Regression Model Summary** (continued)

| | *Dependent variable:* |
|---|---|
| | imdb_score |
| $releaseyear^2$ | 0.098 |
| | (0.754) |
| $releaseyear^3$ | 0.592 |
| | (0.756) |
| $duration^1$ | 7.898*** |
| | (0.912) |
| $duration^2$ | −2.739*** |
| | (0.768) |
| $duration^3$ | 0.092 |
| | (0.743) |
| $duration^4$ | 1.186 |
| | (0.735) |
| $duration^5$ | −1.387* |
| | (0.735) |
| movie budget | −0.000*** |
| | (0.000) |
| nb news articles | 0.0001*** |
| | (0.00001) |
| nb faces | −0.015* |
| | (0.008) |
| movie meter IMDBpro | −0.00000** |
| | (0.00000) |
| action | −0.168*** |
| | (0.045) |
| western | −0.122** |
| | (0.058) |
| sport | 0.173** |
| | (0.079) |
| drama | 0.160*** |
| | (0.041) |
| animation | 0.550*** |
| | (0.170) |
| rated R1 | 0.218*** |
| | (0.036) |
| good cinematographer | 0.388*** |

**Table 5: Linear Regression Model Summary** (continued)

| | *Dependent variable:* |
|---|---|
| | imdb_score |
| | (0.040) |
| good director | 0.922*** |
| | (0.042) |
| Constant | 5.731*** |
| | (0.058) |
| Observations | 1,930 |
| $R^2$ | 0.567 |
| Adjusted $R^2$ | 0.562 |
| Residual Std. Error | 0.728 (df = 1909) |
| F Statistic | 124.907*** (df = 20; 1909) |
| *Note:* | *p<0.1; **p<0.05; ***p<0.01 |

# 4 R Code:

```
IMDB_data =
    read.csv("/Users/marcgarraya/Downloads/IMDB_data_Fall_2023.csv",
    header=TRUE)
attach(IMDB_data)

IMDB_test =
    read.csv("/Users/marcgarraya/Downloads/test_data_IMDB_Fall_2023.csv",
    header=TRUE)
density_plot = density(imdb_score)

plot_score=plot(density_plot, main = "imdb_score")

#looking at director
#Defining what a good score is in order to create a variable for good
    directors

good_score = quantile(imdb_score, 0.50) # scores are a bit skewed and we
    want above average directors
director_scores = aggregate(imdb_score ~ director, data = IMDB_data, FUN
    = mean)
director_scores$good_director = ifelse(director_scores$imdb_score >=
    good_score, 1, 0)
director_scores <- director_scores[, -2]
IMDB_data = merge(IMDB_data, director_scores, by.x = "director", by.y =
    "director", all.x = TRUE)

#Looking at Cinematographer
cinematographer_scores = aggregate(imdb_score ~ cinematographer, data =
    IMDB_data, FUN = mean)
cinematographer_scores$good_cinematographer =
    ifelse(cinematographer_scores$imdb_score >= good_score, 1, 0)
cinematographer_scores <- cinematographer_scores[, -2]
IMDB_data = merge(IMDB_data, cinematographer_scores, by.x =
    "cinematographer", by.y = "cinematographer", all.x = TRUE)

IMDB_data$rated_R <- ifelse(IMDB_data$maturity_ratin == 'R', 1, 0)
#Looking at distributors

table(distributor)
distributor_size = aggregate(movie_title ~ distributor, data =
    IMDB_data, FUN = length)
big_distributor = quantile(distributor_size$movie_title, 0.90)

distributor_size$big_distributor = ifelse(distributor_size$movie_title
    >= big_distributor, 1, 0)
distributor_size <- distributor_size[, -2]
IMDB_data = merge(IMDB_data, distributor_size, by.x = "distributor",
    by.y = "distributor", all.x = TRUE)


#Looking at production company
table(production_company)
```

```
production_company_size = aggregate(movie_title ~ production_company,
    data = IMDB_data, FUN = length)
big_production_company = quantile(production_company_size$movie_title,
    0.90)

production_company_size$big_production_company =
    ifelse(production_company_size$movie_title >= big_production_company,
    1, 0)
production_company_size <- production_company_size[, -2]
IMDB_data = merge(IMDB_data, production_company_size, by.x =
    "production_company", by.y = "production_company", all.x = TRUE)

###Identifying all the variables as categorical variables
IMDB_data$rated_R=as.factor(IMDB_data$rated_R)
IMDB_data$action=as.factor(IMDB_data$action)
IMDB_data$adventure=as.factor(IMDB_data$adventure)
IMDB_data$scifi=as.factor(IMDB_data$scifi)
IMDB_data$thriller=as.factor(IMDB_data$thriller)
IMDB_data$musical=as.factor(IMDB_data$musical)
IMDB_data$romance=as.factor(IMDB_data$romance)
IMDB_data$drama=as.factor(IMDB_data$drama)
IMDB_data$war=as.factor(IMDB_data$war)
IMDB_data$animation=as.factor(IMDB_data$animation)
IMDB_data$crime=as.factor(IMDB_data$crime)
IMDB_data$western=as.factor(IMDB_data$horror)
IMDB_data$sport=as.factor(IMDB_data$sport)
IMDB_data$colour_film=as.factor(IMDB_data$colour_film)
IMDB_data$language=as.factor(IMDB_data$language)
IMDB_data$country=as.factor(IMDB_data$country)
IMDB_data$big_production_company=as.factor(IMDB_data$big_production_company)
IMDB_data$big_distributor=as.factor(IMDB_data$big_distributor)
IMDB_data$good_director=as.factor(IMDB_data$good_director)
IMDB_data$good_cinematographer=as.factor(IMDB_data$good_cinematographer)
attach(IMDB_data)

#imdb_score

summary(imdb_score)

boxplot(imdb_score, col = "blue")
hist(imdb_score, breaks = 20, col = "blue")

#movie_budget

summary(movie_budget)
boxplot(movie_budget, main="Box Plot of Movie Budget", col="purple")
hist(movie_budget, main="Histogram for Movie Budget",
    col="purple",breaks = 20)


#release_day

summary(release_day)

boxplot(release_day)
hist(release_day, breaks = 20)
```

```
#relase_year
summary(release_year)
boxplot(release_year)
hist(release_year, breaks=20)

#duration

summary(duration)
plot(duration)#bizarre le plot, too concentrated en bas
hist(duration)

#aspect_ratio

summary(aspect_ratio)
plot(aspect_ratio)#non y'a deux lignes bizarres
hist(aspect_ratio)#chelou

#aspect_ratio
summary(aspect_ratio)
plot(aspect_ratio)
hist(aspect_ratio)

summary(aspect_ratio)
plot(aspect_ratio)#non y'a deux lignes bizarres
hist(aspect_ratio)#chelou

#nb_news_articles

summary(nb_news_articles)
plot(nb_news_articles)#hein
hist(nb_news_articles)#hein

#install package for hetereoskedasticity
install.packages("car")
library(car)

#imdb_score & movie_budget

lm.budget=lm(imdb_score~movie_budget)
summary(lm.budget)
plot(imdb_score,movie_budget)

   #Heteroskedasticity
residualPlot(lm.budget,quadratic=FALSE)

   #NCV test
ncvTest(lm.budget) #it is heteroskedastic

   #correct heteroskedasticity
install.packages('lmtest')
install.packages('plm')
require(lmtest)
require(plm)
coeftest(lm.budget,vcov=vcovHC(lm.budget,type='HC1'))
```

```
#imdb_score & release_year

lm.year=lm(imdb_score~release_year)
summary(lm.year)
plot(imdb_score,release_year)

    #Heteroskedasticity
residualPlot(lm.year,quadratic=FALSE)#no visual evidence imo

    #NCV test
ncvTest(lm.year)#p-value>0.05 so no heteroskedasticity is present

#imdb_score & duration

lm.duration=lm(imdb_score~duration)
summary(lm.duration)
plot(imdb_score,duration)

   #Heteroskedasticity
residualPlot(lm.duration,quadratic=FALSE)#no visual evidence imo

  #NCV test
ncvTest(lm.duration)#p-value<0.05 so heteroskedasticity is present

  #correct heteroskedasticity
require(lmtest)
require(plm)
coeftest(lm.duration,vcov=vcovHC(lm.duration,type='HC1'))

#imdb_score & nb_news_articles

lm.news=lm(imdb_score~nb_news_articles)
summary(lm.news) #nul
plot(imdb_score,nb_news_articles)#nul

   #Heteroskedasticity
residualPlot(lm.news,quadratic=FALSE)#no visual evidence imo

   #NCV test
ncvTest(lm.news)

   #correct heteroskedasticity
require(lmtest)
require(plm)
coeftest(lm.news,vcov=vcovHC(lm.news,type='HC1'))#oue mais chelou

#imdb_score & nb_faces

lm.faces=lm(imdb_score~nb_faces)
summary(lm.faces) #nul
plot(imdb_score,nb_faces)#ca va pas bcp de outliers
abline(lm.faces)#not a good fit

   #Heteroskedasticity
residualPlot(lm.faces,quadratic=FALSE)#no visual evidence imo
```

```
   #numerical test for heteroskedasticity
ncvTest(lm.faces)# no heteroskedasticity pval>0.05

#imdb_score & movie_meter_IMDBpro
reg_movie_meter=lm(imdb_score~movie_meter_IMDBpro) #Seems Heteroskedastic
summary(reg_movie_meter)
residualPlot(reg_movie_meter, quadratic=FALSE)

   #numerical test of heteroskedasticity
ncvTest(reg_movie_meter) #p val<0.05 so there is heteroskedasticity

   #correct heteroskedasticity
require(lmtest)
require(plm)
coeftest(reg_movie_meter,vcov=vcovHC(reg_movie_meter,type='HC1'))

#imdb_score & actor3_star_meter
reg_actor3_meter=lm(imdb_score~actor3_star_meter)
summary(reg_actor3_meter)
residualPlot(reg_actor3_meter, quadratic=FALSE)

  #numerical test for heteroskedasticity
ncvTest(reg_actor3_meter)

   #correct heteroskedasticity
require(lmtest)
require(plm)
coeftest(reg_actor3_meter,vcov=vcovHC(reg_actor3_meter,type='HC1'))

#imdb_score & actor2_star_meter
reg_actor2_meter=lm(imdb_score~actor2_star_meter) #Not Heteroskedastic
summary(reg_actor2_meter)
residualPlot(reg_actor2_meter, quadratic=FALSE)
ncvTest(reg_actor2_meter)

#imdb_score & actor1_star_meter
reg_actor1_meter=lm(imdb_score~actor1_star_meter) #Not Heteroskedastic
summary(reg_actor1_meter)
residualPlot(reg_actor1_meter, quadratic=FALSE)
ncvTest(reg_actor1_meter)

#Running loops to find best poly degree
install.packages('lmtest')
install.packages('plm')
install.packages("car")
install.packages("rsq")
library(rsq)
library(car)
require(lmtest)
require(plm)
library(boot)


best_degree1 = 1
best_mse1 = NULL
best_model1 = NULL
```

```
for (degree in 1:5) {
  model1 = glm(imdb_score ~ poly(movie_budget, degree, raw = TRUE))
  mse1 = cv.glm(data = IMDB_data, glmfit = model1, K = 5)$delta[1]

  if (is.null(best_mse1) || mse1 < best_mse1) {
    best_degree1 = degree
    best_mse1 = mse1
    best_model1 = model1
  }
}

cat("Best Degree:", best_degree1, "\n")
cat("Best MSE:", best_mse1, "\n")
summary(best_model1)

rsq(best_model1)

#imdb_score & release_year
best_degree2 = 1
best_mse2 = NULL
best_model2 = NULL

for (degree in 1:5) {
  model2 = glm(imdb_score ~ poly(release_year, degree, raw = TRUE))
  mse2 = cv.glm(data = IMDB_data, glmfit = model2, K = 5)$delta[1]

  if (is.null(best_mse2) || mse2 < best_mse2) {
    best_degree2 = degree
    best_mse2 = mse2
    best_model2 = model2
  }
}

cat("Best Degree:", best_degree2, "\n")
cat("Best MSE:", best_mse2, "\n")
summary(best_model2)

rsq(best_model2)

#imdb_score & duration
best_degree3 = 1
best_mse3 = NULL
best_model3 = NULL

for (degree in 1:5) {
  model3 = glm(imdb_score ~ poly(duration, degree, raw = TRUE))
  mse3 = cv.glm(data = IMDB_data, glmfit = model3, K = 5)$delta[1]

  if (is.null(best_mse3) || mse3 < best_mse3) {
    best_degree3 = degree
    best_mse3 = mse3
    best_model3 = model3
  }
}
```

```
cat("Best Degree:", best_degree3, "\n")
cat("Best MSE:", best_mse3, "\n")
summary(best_model3)

rsq(best_model3)

#imdb_score & nb_news_articles
best_degree4 = 1
best_mse4 = NULL
best_model4 = NULL

for (degree in 1:5) {
  model4 = glm(imdb_score ~ poly(nb_news_articles, degree, raw = TRUE))
  mse4 = cv.glm(data = IMDB_data, glmfit = model4, K = 5)$delta[1]

  if (is.null(best_mse4) || mse4 < best_mse4) {
    best_degree4 = degree
    best_mse4 = mse4
    best_model4 = model4
  }
}

cat("Best Degree:", best_degree4, "\n")
cat("Best MSE:", best_mse4, "\n")
summary(best_model4)

rsq(best_model4)
#imdb_score & nb_faces
best_degree5 = 1
best_mse5 = NULL
best_model5 = NULL

for (degree in 1:5) {
  model5 = glm(imdb_score ~ poly(nb_faces, degree, raw = TRUE))
  mse5 = cv.glm(data = IMDB_data, glmfit = model5, K = 5)$delta[1]

  if (is.null(best_mse5) || mse5 < best_mse5) {
    best_degree5 = degree
    best_mse5 = mse5
    best_model5 = model5
  }
}

cat("Best Degree:", best_degree5, "\n")
cat("Best MSE:", best_mse5, "\n")
summary(best_model5)

rsq(best_model5)
#imdb_score & actor1_star_meter
best_degree6 = 1
best_mse6 = NULL
best_model6 = NULL

for (degree in 1:5) {
  model6 = glm(imdb_score ~ poly(actor1_star_meter, degree, raw = TRUE))
```

```
  mse6 = cv.glm(data = IMDB_data, glmfit = model6, K = 5)$delta[1]

  if (is.null(best_mse6) || mse6 < best_mse6) {
    best_degree6 = degree
    best_mse6 = mse6
    best_model6 = model6
  }
}

cat("Best Degree:", best_degree6, "\n")
cat("Best MSE:", best_mse6, "\n")
summary(best_model6)
rsq(best_model6)

#imdb_score & actor2_star_meter
best_degree7 = 1
best_mse7 = NULL
best_model7 = NULL

for (degree in 1:5) {
  model7 = glm(imdb_score ~ poly(actor2_star_meter, degree, raw = TRUE))
  mse7 = cv.glm(data = IMDB_data, glmfit = model7, K = 5)$delta[1]

  if (is.null(best_mse7) || mse7 < best_mse7) {
    best_degree7 = degree
    best_mse7 = mse7
    best_model7 = model7
  }
}

cat("Best Degree:", best_degree7, "\n")
cat("Best MSE:", best_mse7, "\n")
summary(best_model7)

rsq(best_model7)
#imdb_score & actor3_star_meter
best_degree8 = 1
best_mse8 = NULL
best_model8 = NULL

for (degree in 1:5) {
  model8 = glm(imdb_score ~ poly(actor3_star_meter, degree, raw = TRUE))
  mse8 = cv.glm(data = IMDB_data, glmfit = model8, K = 5)$delta[1]

  if (is.null(best_mse8) || mse8 < best_mse8) {
    best_degree8 = degree
    best_mse8 = mse8
    best_model8 = model8
  }
}

cat("Best Degree:", best_degree8, "\n")
cat("Best MSE:", best_mse8, "\n")
summary(best_model8)
rsq(best_model8)
#plot each variable against imdb_score
```

```
attach(IMDB_data)
install.packages("ggplot2")
library(ggplot2)

#looking at qualitative variable R^2

lm_good_director = lm(imdb_score~good_director)
summary(lm_good_director)

lm_good_cinematographer = lm(imdb_score~good_cinematographer)
summary(lm_good_cinematographer)

lm_big_distributor = lm(imdb_score~IMDB_data$big_distributor)
summary(lm_big_distributor)

lm_big_production_company =
    lm(imdb_score~IMDB_data$big_production_company)
summary(lm_big_production_company)

lm_genres =
    lm(imdb_score~action+adventure+scifi+thriller+musical+romance+western+sport+horror+dr
summary(lm_genres)

lm_rated_R = lm(imdb_score~rated_R)
summary(lm_rated_R)

#movie_budget
plot_movie_budget = ggplot(IMDB_data, aes(y=imdb_score, x=movie_budget))
scatter=geom_point(color="grey")
plot_movie_budget+scatter

#release_year
plot_release_year = ggplot(IMDB_data, aes(y=imdb_score, x=release_year))
scatter=geom_point(color="grey")
plot_release_year+scatter

#Duration
plot_duration = ggplot(IMDB_data, aes(y=imdb_score, x=duration))
scatter=geom_point(color="grey")
plot_duration+scatter

#maturity_rating
plot_rated_R = ggplot(IMDB_data, aes(y=imdb_score, x=rated_R))
scatter=geom_point(color="grey")
plot_rated_R+scatter

#actor1_star_meter
plot_actor1_star_meter = ggplot(IMDB_data, aes(y=imdb_score,
    x=actor1_star_meter))
scatter=geom_point(color="grey")
plot_actor1_star_meter+scatter

#actor2_star_meter
plot_actor2_star_meter = ggplot(IMDB_data, aes(y=imdb_score,
    x=actor2_star_meter))
```

```
scatter=geom_point(color="grey")
plot_actor2_star_meter+scatter

#actor3_star_meter
plot_actor3_star_meter = ggplot(IMDB_data, aes(y=imdb_score,
    x=actor3_star_meter))
scatter=geom_point(color="grey")
plot_actor3_star_meter+scatter

#action
plot_action = ggplot(IMDB_data, aes(y=imdb_score, x=action))
scatter=geom_point(color="grey")
plot_action+scatter

#movie_meter_IMDBpro
plot_movie_meter_IMDBpro = ggplot(IMDB_data, aes(y=imdb_score,
    x=movie_meter_IMDBpro))
scatter=geom_point(color="grey")
plot_movie_meter_IMDBpro+scatter

#Trying models

numeric_vars <- c("movie_budget", "release_day", "release_year",
    "duration", "nb_news_articles", "nb_faces", "actor1_star_meter",
    "actor2_star_meter", "actor3_star_meter",
    "movie_meter_IMDBpro","action","adventure","scifi","thriller","musical","romance","we
    "rated_R", "good_director", "good_cinematographer",
    "big_distributor", "big_production_company")

# Convert selected columns to numeric
IMDB_data[numeric_vars] <- lapply(IMDB_data[numeric_vars], as.numeric)

# Calculate the correlation matrix
cor_matrix <- cor(IMDB_data[numeric_vars])

# Print the correlation matrix
print(cor_matrix)


# Define the threshold for high correlation
correlation_threshold <- 0.8

# Create a logical matrix indicating correlations above the threshold
high_correlation_matrix <- abs(cor_matrix) > correlation_threshold &
    cor_matrix != 1

# Find variable pairs with high correlations
high_correlation_pairs <- which(high_correlation_matrix, arr.ind = TRUE)

# Print the variable pairs and their correlations
for (pair in high_correlation_pairs) {
  variable1 <- rownames(cor_matrix)[pair[1]]
  variable2 <- colnames(cor_matrix)[pair[2]]
  correlation <- cor_matrix[pair[1], pair[2]]
  cat(sprintf("Correlation between %s and %s: %.2f\n", variable1,
      variable2, correlation))
```

```
}


### Backward Elimination
install.packages("tidyverse")
install.packages("leaps")
library(tidyverse)
library(caret)
library(leaps)
library(MASS)

backward = IMDB_data[numeric_vars]

models <- regsubsets(imdb_score~., data = backward, nvmax = 15,
                     method = "backward")
best_model <- summary(models)$which[which.max(summary(models)$adjr2),]

summary(models)
#Running a regression with all good variables and their ideal poly degree

degrees =  c(1, 2, 3, 4, 5)
best_combination = c()
best_mse = Inf
K = 5

for (a in degrees) {
  for (b in degrees) {
    for (c in degrees) {
      for (d in degrees) {
        for (e in degrees) {
          for (f in degrees) {

          poly_year = poly(IMDB_data$release_year, degree = a, raw = TRUE)
          poly_duration = poly(IMDB_data$duration, degree = b, raw = TRUE)
          poly_budget = poly(IMDB_data$movie_budget, degree = c, raw =
              TRUE)
          poly_news = poly(IMDB_data$nb_news_articles, degree = d, raw =
              TRUE)
          poly_faces = poly(IMDB_data$nb_faces, degree = e, raw = TRUE)
          poly_IMDBpro = poly(IMDB_data$movie_meter_IMDBpro, degree = f,
              raw = TRUE)
          lm_action = poly(IMDB_data$action, degree = 1, raw = TRUE)
          lm_western = poly(IMDB_data$western, degree = 1, raw = TRUE)
          lm_sport = poly(IMDB_data$sport, degree = 1, raw = TRUE)
          lm_drama = poly(IMDB_data$drama, degree = 1, raw = TRUE)
          lm_animation = poly(IMDB_data$animation, degree = 1, raw = TRUE)
          lm_rated = poly(IMDB_data$rated_R, degree = 1, raw = TRUE)
          lm_cinematographer = poly(IMDB_data$good_cinematographer, degree
              = 1, raw = TRUE)
          lm_director = poly(IMDB_data$good_director, degree = 1, raw =
              TRUE)

          formula = as.formula(paste("imdb_score ~",
                                     paste("action", "poly_year",
                                         "poly_duration", "poly_budget",
                                         "poly_news","poly_faces",
```

```
                                        "poly_IMDBpro", "lm_action",
                                        "lm_western", "lm_sport",
                                        "lm_drama", "lm_animation",
                                        "lm_rated", "lm_cinematographer",
                                        "lm_director", sep = " + ")
        ))

        poly_data = data.frame(poly_year, poly_duration, poly_budget,
            poly_news, poly_faces, poly_IMDBpro,lm_action,lm_western,
            lm_sport, lm_drama,lm_animation,lm_rated, lm_cinematographer,
            lm_director)

        glm_model = glm(formula, data = poly_data)
        cv_result = cv.glm(data = poly_data, glmfit = glm_model, K = K)
        mse = mean(cv_result$delta)

        if (mse < best_mse) {
          best_combination = c(a, b, c, d, e, f)
          best_mse = mse
            }
          }
        }
      }
    }
  }
}

cat("Best combination of polynomial degrees (a, b, c, d, e, f):",
    best_combination, "\n")
cat("Lowest MSE:", best_mse, "\n")


lm_all = lm(imdb_score~poly(release_year, 3)+poly(duration,
    5)+movie_budget+nb_news_articles+nb_faces+movie_meter_IMDBpro+action+western+sport+dr
summary(lm_all)

glm_all = glm(imdb_score~poly(release_year, 3)+poly(duration,
    5)+movie_budget+nb_news_articles+nb_faces+movie_meter_IMDBpro+action+western+sport+dr
coeftest(glm_all,vcov=vcovHC(glm_all,type='HC1'))
mse = cv.glm(data = IMDB_data, glmfit = glm_all, K = 20)$delta[1]
print(mse)
rsq(glm_all)


#Cross-validation in a multiple spline model

y1 = quantile(release_year, 0.25)
y2 = quantile(release_year, 0.5)
y3 = quantile(release_year, 0.75)

d1 = quantile(duration, 0.25)
d2 = quantile(duration, 0.5)
d3 = quantile(duration, 0.75)

a1 = quantile(duration, 0.25)
a2 = quantile(duration, 0.5)
```

```
a3 = quantile(duration, 0.75)

library(splines)

fit=glm(imdb_score~bs(release_year, knots = c(y1,y2,y3) , degree=3)+
          bs(duration, knots = c(d1,d2,d3) , degree=5)+
          bs(nb_news_articles, knots = c(a1,a2,a3), degree=1)+
          good_cinematographer+good_director+action+adventure+scifi+thriller+musical+rom
summary(fit)
rsq(fit)
mse = cv.glm(data = IMDB_data, glmfit = fit, K = 20)$delta[1]
print(mse)

#We are keeping the first model as it has a lower mse

install.packages("stargazer")
install.packages("xtable")
library(xtable)
library(stargazer)

latex_code <- stargazer(lm_all, type = "latex", title = "Linear
    Regression Model Summary")


#Using model to predict test
IMDB_test = merge(IMDB_test, director_scores, by.x = "director", by.y =
    "director", all.x = TRUE)
IMDB_test$good_director[is.na(IMDB_test$good_director)] = 0
IMDB_test = merge(IMDB_test, cinematographer_scores, by.x =
    "cinematographer", by.y = "cinematographer", all.x = TRUE)
IMDB_test$good_cinematographer[is.na(IMDB_test$good_cinematographer)] = 0

IMDB_test$rated_R <- ifelse(IMDB_test$maturity_ratin == 'R', 1, 0)


IMDB_test$rated_R=as.factor(IMDB_test$rated_R)
IMDB_test$good_director=as.factor(IMDB_test$good_director)
IMDB_test$good_cinematographer=as.factor(IMDB_test$good_cinematographer)
IMDB_test$action=as.factor(IMDB_test$action)
IMDB_test$adventure=as.factor(IMDB_test$adventure)
IMDB_test$scifi=as.factor(IMDB_test$scifi)
IMDB_test$thriller=as.factor(IMDB_test$thriller)
IMDB_test$musical=as.factor(IMDB_test$musical)
IMDB_test$romance=as.factor(IMDB_test$romance)
IMDB_test$drama=as.factor(IMDB_test$drama)
IMDB_test$war=as.factor(IMDB_test$war)
IMDB_test$animation=as.factor(IMDB_test$animation)
IMDB_test$crime=as.factor(IMDB_test$crime)
IMDB_test$western=as.factor(IMDB_test$horror)
IMDB_test$sport=as.factor(IMDB_test$sport)
IMDB_test$colour_film=as.factor(IMDB_test$colour_film)
IMDB_test$language=as.factor(IMDB_test$language)
IMDB_test$country=as.factor(IMDB_test$country)

attach(IMDB_test)
```

```
IMDB_test$movie_budget <- as.numeric(gsub(",", "",
    IMDB_test$movie_budget))
IMDB_test$movie_budget <- as.numeric(IMDB_test$movie_budget)
IMDB_test$predicted_imdb_scores = predict(glm_all, newdata = IMDB_test)
movie_names_and_scores <- IMDB_test[c("movie_title",
    "predicted_imdb_scores")]
print(movie_names_and_scores)
```