Final project

Boyu Wu

Due: December 19 2021

Executive Summary:

Problem In this final project, we want to analyze the effects of data on the lifetime gross of a movie. **Data** We will try to predict what metadata influences the money-making aspect the most. This data we extracted from multiple sources:

main data - movielens
imbd data
boxoffice

Analysis Especially we will do use linear regression methods and trees together with ensemble methods to see how well we can predict the response. And look whether a prediction of this kind is even possible. Our analysis of this is being conducted as a regression problem. So that if someone wants they can also use our code and predict the gross. **Conclusion** In the end, we managed to tune an ensemble method that predicts almost 50% of the variance, which can be seen as quite good in terms of prediction. Here is our github repo

Introduction

Background information The context of this project is to run a regression program that can predict the gross of a movie without seeing it. So just by looking at movie review averages, how many people revised it, runtime,we want to predict the earning. This way we can help the movie industry to gain a first impression of how well the movie will perform based on metadata. Through this process, we can also help them to reduce the time that they need to see and damage control stuff in the real world, but they can directly release the movie being sure that it will perform or not perform according to the predictions. **Analysis goal** The goal is to obtain a good model that can accurately depict the dependencies of metadata and lifetime gross. But this model should also be easily explainable and logical. Thus, we opted against doing a nural network. We will not only use features such as runtime, ratings, release year, but we will also onehot encode different genres. As we know there are some genres that generally perform better than others. Success will be evaluated based on either r squared or RSME. **Significance** We need to predict the dependencies accurately and then we only then can put our product (the code) on the market and help the movie industry with their new film releases.

we decided to download the data instead of downloading from a url because some files were extremely large and it took too much time to do it every time from the internet. some of the files can be found in the github files but some are too large to upload them.

```
#due to te fact that movie data is split among many different
#files we need to read in the individual files
set.seed(11)

movie_raw = read.csv("movies.csv")
rating_raw_grouplense = read.csv("ratings.csv")
rating_raw_imbd = read_tsv("data.tsv",show_col_types = FALSE)
links = read.csv("links.csv")
boxoffice = read.csv("boxoffice.csv")
runtime = read_tsv("runtime.tsv",show_col_types = FALSE)
title = read_tsv("title.tsv",show_col_types = FALSE)


#transfor it into the datastructure we are familiar with

movie_raw = movie_raw %>% as_tibble
rating_raw_grouplense = rating_raw_grouplense %>% as_tibble
rating_raw_imbd = rating_raw_imbd %>% as_tibble
links = links %>% as_tibble
boxoffice = boxoffice %>% as_tibble
runtime = runtime %>% as_tibble
title = title %>% as_tibble
```

Links: Identifiers that can be used to link to other sources of movie data are contained in the file links.csv. Each line of this file after the header row represents one movie. Allows us to merge our main data with data (in our case) from imdb. One imporatnt factor is we need to mutate the link so that we can join the data in the end because our imdbId requires a tt followed by a 7 digit number, we need to tidy our data so that this is possilbe.

```
links %>% arrange(imdbId) #print links tibble
```

```
## # A tibble: 62,423 x 3
##    movieId imdbId tmdbId
##      <int>  <int>  <int>
##  1  172063      1  16612
##  2  140539      3  88013
##  3  180695      7 159895
##  4   88674      8 105158
##  5  120869     10    774
##  6   98981     12    160
##  7  199706     13 208234
##  8  113048     14  82120
##  9  199716     16 129436
## 10  199708     22 127777
## # ... with 62,413 more rows
```

```
links$imdbId = sprintf("%07d",links$imdbId)
links = links %>%
  mutate(imdbId = paste("tt",
                        imdbId,
                        collapse = NULL,
                        sep = ""))
```

```
links
```

```
## # A tibble: 62,423 x 3
##    movieId imdbId    tmdbId
##      <int> <chr>      <int>
##  1       1 tt0114709    862
##  2       2 tt0113497   8844
##  3       3 tt0113228  15602
##  4       4 tt0114885  31357
##  5       5 tt0113041  11862
##  6       6 tt0113277    949
##  7       7 tt0114319  11860
##  8       8 tt0112302  45325
##  9       9 tt0114576   9091
## 10      10 tt0113189    710
## # ... with 62,413 more rows
```

Runtime: this is one of our bigger datasets with a few interesting things we want to extract. Among others, tconst for merging and isAdult and runtimeMinutes for data analysis (we want to use as many features we can get our hands on as possible (ofcourse these features need to have an influence over our predicted boxoffice results).

```
runtime = runtime %>%
    mutate_at(
      vars(tconst, isAdult,runtimeMinutes),
      na_if, "\\N") %>%
  select(
    c("tconst","isAdult","runtimeMinutes")) %>%
  drop_na()
runtime
```

```
## # A tibble: 2,322,790 x 3
##    tconst    isAdult runtimeMinutes
##    <chr>       <dbl> <chr>
##  1 tt0000001       0 1
##  2 tt0000002       0 5
##  3 tt0000003       0 4
##  4 tt0000004       0 12
##  5 tt0000005       0 1
##  6 tt0000006       0 1
##  7 tt0000007       0 1
##  8 tt0000008       0 1
##  9 tt0000009       0 40
## 10 tt0000010       0 1
## # ... with 2,322,780 more rows
```

Boxoffice: Lifetime_gross is included in the boxoffice.csv. This data represents the best >15k grossing films all time till 2019 - originally this data is parsed from Boxofficemojo.

```
boxoffice = boxoffice %>%                    #filter boxoffice data
  mutate(wtitle = str_to_lower(title)) %>%
  select(c("rank","title","lifetime_gross"))
boxoffice                                    #print out boxoffice
```

```
## # A tibble: 16,542 x 3
##     rank title                         lifetime_gross
##    <int> <chr>                                  <int>
##  1     1 Star Wars: The Force Awakens      936662225
##  2     2 Avengers: Endgame                 857190335
##  3     3 Avatar                            760507625
##  4     4 Black Panther                     700059566
##  5     5 Avengers: Infinity War            678815482
##  6     6 Titanic                           659363944
##  7     7 Jurassic World                    652270625
##  8     8 Marvel's The Avengers             623357910
##  9     9 Star Wars: The Last Jedi          620181382
## 10    10 Incredibles 2                     608581744
## # ... with 16,532 more rows
```

Movie_raw: Movie information is contained in the file movies.csv. Each line of this file after the header row represents one movie. This is our main "frame". This data we still need to tidy up: -we have release year and title in one and we need to one hot encode genres.

```
movie_raw
```

```
## # A tibble: 62,423 x 3
##    movieId title                              genres
##      <int> <chr>                              <chr>
##  1       1 Toy Story (1995)                   Adventure|Animation|Children|Come~
##  2       2 Jumanji (1995)                     Adventure|Children|Fantasy
##  3       3 Grumpier Old Men (1995)            Comedy|Romance
##  4       4 Waiting to Exhale (1995)           Comedy|Drama|Romance
##  5       5 Father of the Bride Part II (1995) Comedy
##  6       6 Heat (1995)                        Action|Crime|Thriller
##  7       7 Sabrina (1995)                     Comedy|Romance
##  8       8 Tom and Huck (1995)                Adventure|Children
##  9       9 Sudden Death (1995)                Action
## 10      10 GoldenEye (1995)                   Action|Adventure|Thriller
## # ... with 62,413 more rows
```

```
movie_raw = movie_raw %>%
  separate(title,                                    #seperate release year
           into = c("title","year"),                 #and movie title
           sep = "\\(" ) %>%
  separate(year,into = c("year", "xxx"), sep = "\\)") %>%
  select(-xxx) %>%
  filter(str_detect(year,"^[1-9][0-9][0-9][0-9]"))%>%#regex pattern matching
  mutate(year = as.integer(year))
movie_raw = movie_raw %>% mutate(year = as.integer(year))

genres_df = movie_raw %>% mutate(value = 1) %>%      #one hot encoding for genre
  separate_rows(genres, sep = "\\|") %>%
  spread(genres, value, fill =0)


genres_df = genres_df %>%                            #mutate a lowercase title
  mutate(wtitle = str_to_lower(title))
```

```
boxoffice = boxoffice %>%                              #mutate a lowercase title
  mutate(wtitle = str_to_lower(title))

genres_df
```

```
## # A tibble: 57,244 x 24
##    movieId title       year `(no genres lis~ Action Adventure Animation Children
##      <int> <chr>      <int>            <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
##  1       1 "Toy Stor~  1995                0      0         1         1        1
##  2       2 "Jumanji "  1995                0      0         1         0        1
##  3       3 "Grumpier~  1995                0      0         0         0        0
##  4       4 "Waiting ~  1995                0      0         0         0        0
##  5       5 "Father o~  1995                0      0         0         0        0
##  6       6 "Heat "     1995                0      1         0         0        0
##  7       7 "Sabrina "  1995                0      0         0         0        0
##  8       8 "Tom and ~  1995                0      0         1         0        1
##  9       9 "Sudden D~  1995                0      1         0         0        0
## 10      10 "GoldenEy~  1995                0      1         1         0        0
## # ... with 57,234 more rows, and 16 more variables: Comedy <dbl>, Crime <dbl>,
## #   Documentary <dbl>, Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>,
## #   Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>, Romance <dbl>,
## #   Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>, wtitle <chr>
```

```
genres_df
```

```
## # A tibble: 57,244 x 24
##    movieId title       year `(no genres lis~ Action Adventure Animation Children
##      <int> <chr>      <int>            <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
##  1       1 "Toy Stor~  1995                0      0         1         1        1
##  2       2 "Jumanji "  1995                0      0         1         0        1
##  3       3 "Grumpier~  1995                0      0         0         0        0
##  4       4 "Waiting ~  1995                0      0         0         0        0
##  5       5 "Father o~  1995                0      0         0         0        0
##  6       6 "Heat "     1995                0      1         0         0        0
##  7       7 "Sabrina "  1995                0      0         0         0        0
##  8       8 "Tom and ~  1995                0      0         1         0        1
##  9       9 "Sudden D~  1995                0      1         0         0        0
## 10      10 "GoldenEy~  1995                0      1         1         0        0
## # ... with 57,234 more rows, and 16 more variables: Comedy <dbl>, Crime <dbl>,
## #   Documentary <dbl>, Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>,
## #   Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>, Romance <dbl>,
## #   Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>, wtitle <chr>
```

```
genres_df_big = genres_df %>%
  inner_join(links, by = c("movieId"))             #testing our inner join
```

We realize that something is wrong! We obviously have way too little data in the merged tibble. One possibility was that it has something to do with lower and uppercase, but after implementing lowercase for all, this still hasnt fixed the problem. If we look into the our genre data we will realize that we can only extract the whole row if we add a space after the last character. Thus we removed the last space to check again. -> luckily for us this solved the merging problem.

```
genres_df
```

```
## # A tibble: 57,244 x 24
##    movieId title       year `(no genres lis~ Action Adventure Animation Children
##      <int> <chr>      <int>            <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
##  1       1 "Toy Stor~  1995                0      0         1         1        1
##  2       2 "Jumanji "  1995                0      0         1         0        1
##  3       3 "Grumpier~  1995                0      0         0         0        0
##  4       4 "Waiting ~  1995                0      0         0         0        0
##  5       5 "Father o~  1995                0      0         0         0        0
##  6       6 "Heat "     1995                0      1         0         0        0
##  7       7 "Sabrina "  1995                0      0         0         0        0
##  8       8 "Tom and ~  1995                0      0         1         0        1
##  9       9 "Sudden D~  1995                0      1         0         0        0
## 10      10 "GoldenEy~  1995                0      1         1         0        0
## # ... with 57,234 more rows, and 16 more variables: Comedy <dbl>, Crime <dbl>,
## #   Documentary <dbl>, Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>,
## #   Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>, Romance <dbl>,
## #   Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>, wtitle <chr>
```

```
#demonstration code
genres_df %>% filter(title == "Toy Story")
```

```
## # A tibble: 0 x 24
## # ... with 24 variables: movieId <int>, title <chr>, year <int>,
## #   (no genres listed) <dbl>, Action <dbl>, Adventure <dbl>, Animation <dbl>,
## #   Children <dbl>, Comedy <dbl>, Crime <dbl>, Documentary <dbl>, Drama <dbl>,
## #   Fantasy <dbl>, Film-Noir <dbl>, Horror <dbl>, IMAX <dbl>, Musical <dbl>,
## #   Mystery <dbl>, Romance <dbl>, Sci-Fi <dbl>, Thriller <dbl>, War <dbl>,
## #   Western <dbl>, wtitle <chr>
```

```
genres_df %>% filter(title == "Toy Story ")
```

```
## # A tibble: 1 x 24
##   movieId title      year `(no genres liste~ Action Adventure Animation Children
##     <int> <chr>     <int>              <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
## 1       1 "Toy Sto~  1995                  0      0         1         1        1
## # ... with 16 more variables: Comedy <dbl>, Crime <dbl>, Documentary <dbl>,
## #   Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>, Horror <dbl>, IMAX <dbl>,
## #   Musical <dbl>, Mystery <dbl>, Romance <dbl>, Sci-Fi <dbl>, Thriller <dbl>,
## #   War <dbl>, Western <dbl>, wtitle <chr>
```

```
genres_df
```

```
## # A tibble: 57,244 x 24
##    movieId title       year `(no genres lis~ Action Adventure Animation Children
##      <int> <chr>      <int>            <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
##  1       1 "Toy Stor~  1995                0      0         1         1        1
##  2       2 "Jumanji "  1995                0      0         1         0        1
##  3       3 "Grumpier~  1995                0      0         0         0        0
##  4       4 "Waiting ~  1995                0      0         0         0        0
```

```
##  5       5 "Father o~  1995                0      0         0         0        0
##  6       6 "Heat "     1995                0      1         0         0        0
##  7       7 "Sabrina "  1995                0      0         0         0        0
##  8       8 "Tom and ~  1995                0      0         1         0        1
##  9       9 "Sudden D~  1995                0      1         0         0        0
## 10      10 "GoldenEy~  1995                0      1         1         0        0
## # ... with 57,234 more rows, and 16 more variables: Comedy <dbl>, Crime <dbl>,
## #   Documentary <dbl>, Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>,
## #   Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>, Romance <dbl>,
## #   Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>, wtitle <chr>
```

```
boxoffice
```

```
## # A tibble: 16,542 x 4
##     rank title                         lifetime_gross wtitle
##    <int> <chr>                                  <int> <chr>
##  1     1 Star Wars: The Force Awakens      936662225 star wars: the force awake~
##  2     2 Avengers: Endgame                 857190335 avengers: endgame
##  3     3 Avatar                            760507625 avatar
##  4     4 Black Panther                     700059566 black panther
##  5     5 Avengers: Infinity War            678815482 avengers: infinity war
##  6     6 Titanic                           659363944 titanic
##  7     7 Jurassic World                    652270625 jurassic world
##  8     8 Marvel's The Avengers             623357910 marvel's the avengers
##  9     9 Star Wars: The Last Jedi          620181382 star wars: the last jedi
## 10    10 Incredibles 2                     608581744 incredibles 2
## # ... with 16,532 more rows
```

```
genres_df = genres_df %>%
  mutate(wtitle = across(where(is.character),
                         str_remove_all,
                         pattern = fixed(" "))[[2]]) #uniformly cleaning of data
boxoffice = boxoffice %>%
  mutate(wtitle = across(where(is.character),
                         str_remove_all,
                         pattern = fixed(" "))[[2]])%>%                          #mutate a lowercase
  mutate(wtitle = str_to_lower(wtitle)) #uniformly cleaning of data
boxoffice = genres_df %>% select(-title) %>% inner_join(boxoffice, by = c("wtitle"))
```

Rating_grouplense: All ratings are contained in the file ratings.csv. Each line of this file after the header row represents one rating of one movie by one user. This is one of the ratings we will make use of.

```
rating_grouplense = rating_raw_grouplense %>% select(c("movieId","rating")) %>% group_by(movieId) %>% su
rating_grouplense
```

```
## # A tibble: 59,047 x 2
##    movieId avg_rating_grouplense
##      <int>                 <dbl>
##  1       1                  3.89
##  2       2                  3.25
##  3       3                  3.14
```

```
##  4       4                  2.85
##  5       5                  3.06
##  6       6                  3.85
##  7       7                  3.36
##  8       8                  3.11
##  9       9                  2.99
## 10      10                  3.42
## # ... with 59,037 more rows
```

Here we join the raw imbd data with our runtime data.

```
rating_raw_imbd
```

```
## # A tibble: 1,197,424 x 3
##    tconst    averageRating numVotes
##    <chr>             <dbl>    <dbl>
##  1 tt0000001           5.7     1846
##  2 tt0000002           6        236
##  3 tt0000003           6.5     1606
##  4 tt0000004           6        153
##  5 tt0000005           6.2     2429
##  6 tt0000006           5.2      158
##  7 tt0000007           5.4      760
##  8 tt0000008           5.5     1992
##  9 tt0000009           5.9      191
## 10 tt0000010           6.9     6643
## # ... with 1,197,414 more rows
```

```
imbd_data = rating_raw_imbd %>% left_join(runtime, by = c("tconst"))



#movie_raw %>% left_join(data_2, by = c("column_1","column_2"))
```

Now we will arrange the data by numVotes to verify our plausabilty of the data.

```
imbd_data = imbd_data %>% filter(!isAdult>1) %>% arrange(desc(numVotes))

imbd_data
```

```
## # A tibble: 863,933 x 5
##    tconst    averageRating numVotes isAdult runtimeMinutes
##    <chr>             <dbl>    <dbl>   <dbl> <chr>
##  1 tt0111161           9.3  2508157       0 142
##  2 tt0468569           9    2457982       0 152
##  3 tt1375666           8.8  2204960       0 148
##  4 tt0137523           8.8  1972723       0 139
##  5 tt0109830           8.8  1935587       0 142
##  6 tt0110912           8.9  1934524       0 154
##  7 tt0944947           9.2  1916135       0 57
##  8 tt0133093           8.7  1792408       0 136
##  9 tt0120737           8.8  1753350       0 178
## 10 tt0167260           8.9  1732260       0 201
## # ... with 863,923 more rows
```

```
boxoffice
```

```
## # A tibble: 12,185 x 26
##    movieId  year `(no genres listed)` Action Adventure Animation Children Comedy
##      <int> <int>                <dbl>  <dbl>     <dbl>     <dbl>    <dbl>  <dbl>
##  1       1  1995                    0      0         1         1        1      1
##  2       2  1995                    0      0         1         0        1      0
##  3       3  1995                    0      0         0         0        0      1
##  4       4  1995                    0      0         0         0        0      1
##  5       5  1995                    0      0         0         0        0      1
##  6       6  1995                    0      1         0         0        0      0
##  7       7  1995                    0      0         0         0        0      1
##  8       8  1995                    0      0         1         0        1      0
##  9       9  1995                    0      1         0         0        0      0
## 10      10  1995                    0      1         1         0        0      0
## # ... with 12,175 more rows, and 18 more variables: Crime <dbl>,
## #   Documentary <dbl>, Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>,
## #   Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>, Romance <dbl>,
## #   Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>, wtitle <chr>,
## #   rank <int>, title <chr>, lifetime_gross <int>
```

```
title = title %>%
  mutate(wtitle = across(where(is.character),
                         str_remove_all,
                         pattern = fixed(" "))[[2]])%>%
  mutate(wtitle = str_to_lower(wtitle)) %>% select(-title) %>% filter(isOriginalTitle == 1)
title = title %>% filter(isOriginalTitle == 1)


merge1 = boxoffice %>%  #merge1 is the merge with our titles and our boxoffice results
  left_join(title, by = c("wtitle")) %>% distinct()
merge1
```

```
## # A tibble: 115,423 x 28
##    movieId  year `(no genres listed)` Action Adventure Animation Children Comedy
##      <int> <int>                <dbl>  <dbl>     <dbl>     <dbl>    <dbl>  <dbl>
##  1       1  1995                    0      0         1         1        1      1
##  2       1  1995                    0      0         1         1        1      1
##  3       1  1995                    0      0         1         1        1      1
##  4       2  1995                    0      0         1         0        1      0
##  5       2  1995                    0      0         1         0        1      0
##  6       2  1995                    0      0         1         0        1      0
##  7       2  1995                    0      0         1         0        1      0
##  8       2  1995                    0      0         1         0        1      0
##  9       3  1995                    0      0         0         0        0      1
## 10       4  1995                    0      0         0         0        0      1
## # ... with 115,413 more rows, and 20 more variables: Crime <dbl>,
## #   Documentary <dbl>, Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>,
## #   Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>, Romance <dbl>,
## #   Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>, wtitle <chr>,
## #   rank <int>, title <chr>, lifetime_gross <int>, titleId <chr>,
## #   isOriginalTitle <dbl>
```

We are now realizing a small problem lets take black panther there are 2 same movies that are flagged as originals, this is the due to the fact that one of them was the 1996 version and the other one the 2018 movie that some of us watched in theaters. This problem is a one we cant easily fix but well try in the follwing to minimize the effect by dropping as many duplicates as possible

```
imbd_data = imbd_data %>% mutate(titleId = tconst) %>% select(-tconst)
imbd_data
```

```
## # A tibble: 863,933 x 5
##    averageRating numVotes isAdult runtimeMinutes titleId
##            <dbl>    <dbl>   <dbl> <chr>          <chr>
##  1           9.3  2508157       0 142            tt0111161
##  2           9    2457982       0 152            tt0468569
##  3           8.8  2204960       0 148            tt1375666
##  4           8.8  1972723       0 139            tt0137523
##  5           8.8  1935587       0 142            tt0109830
##  6           8.9  1934524       0 154            tt0110912
##  7           9.2  1916135       0 57             tt0944947
##  8           8.7  1792408       0 136            tt0133093
##  9           8.8  1753350       0 178            tt0120737
## 10           8.9  1732260       0 201            tt0167260
## # ... with 863,923 more rows
```

```
merge2 = merge1 %>% inner_join(imbd_data, by = c("titleId")) #first merge to add avarageRating (imdb nu
links = links %>% mutate(titleId = imdbId) %>% select(-imdbId)
merge2
```

```
## # A tibble: 37,761 x 32
##    movieId  year `(no genres listed)` Action Adventure Animation Children Comedy
##      <int> <int>                <dbl>  <dbl>     <dbl>     <dbl>    <dbl>  <dbl>
##  1       1  1995                    0      0         1         1        1      1
##  2       1  1995                    0      0         1         1        1      1
##  3       2  1995                    0      0         1         0        1      0
##  4       2  1995                    0      0         1         0        1      0
##  5       3  1995                    0      0         0         0        0      1
##  6       4  1995                    0      0         0         0        0      1
##  7       5  1995                    0      0         0         0        0      1
##  8       6  1995                    0      1         0         0        0      0
##  9       6  1995                    0      1         0         0        0      0
## 10       6  1995                    0      1         0         0        0      0
## # ... with 37,751 more rows, and 24 more variables: Crime <dbl>,
## #   Documentary <dbl>, Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>,
## #   Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>, Romance <dbl>,
## #   Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>, wtitle <chr>,
## #   rank <int>, title <chr>, lifetime_gross <int>, titleId <chr>,
## #   isOriginalTitle <dbl>, averageRating <dbl>, numVotes <dbl>, isAdult <dbl>,
## #   runtimeMinutes <chr>
```

```
merge3 = merge2 %>% select(-movieId) %>% inner_join(links, by = c("titleId")) #adding our link data int
merge3
```

```
## # A tibble: 15,984 x 33
##     year `(no genres listed)` Action Adventure Animation Children Comedy Crime
##    <int>                <dbl>  <dbl>     <dbl>     <dbl>    <dbl>  <dbl> <dbl>
##  1  1995                    0      0         1         1        1      1     0
##  2  1995                    0      0         1         0        1      0     0
##  3  1995                    0      0         0         0        0      1     0
##  4  1995                    0      0         0         0        0      1     0
##  5  1995                    0      0         0         0        0      1     0
##  6  1995                    0      1         0         0        0      0     1
##  7  1995                    0      1         0         0        0      0     1
##  8  1995                    0      1         0         0        0      0     1
##  9  1995                    0      0         0         0        0      1     0
## 10  1995                    0      0         0         0        0      1     0
## # ... with 15,974 more rows, and 25 more variables: Documentary <dbl>,
## #   Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>, Horror <dbl>, IMAX <dbl>,
## #   Musical <dbl>, Mystery <dbl>, Romance <dbl>, Sci-Fi <dbl>, Thriller <dbl>,
## #   War <dbl>, Western <dbl>, wtitle <chr>, rank <int>, title <chr>,
## #   lifetime_gross <int>, titleId <chr>, isOriginalTitle <dbl>,
## #   averageRating <dbl>, numVotes <dbl>, isAdult <dbl>, runtimeMinutes <chr>,
## #   movieId <int>, tmdbId <int>
```

```
merge4 = merge3 %>% left_join(rating_grouplense, by = c("movieId")) #we now add grouplense rating into

merge4
```

```
## # A tibble: 15,984 x 34
##     year `(no genres listed)` Action Adventure Animation Children Comedy Crime
##    <int>                <dbl>  <dbl>     <dbl>     <dbl>    <dbl>  <dbl> <dbl>
##  1  1995                    0      0         1         1        1      1     0
##  2  1995                    0      0         1         0        1      0     0
##  3  1995                    0      0         0         0        0      1     0
##  4  1995                    0      0         0         0        0      1     0
##  5  1995                    0      0         0         0        0      1     0
##  6  1995                    0      1         0         0        0      0     1
##  7  1995                    0      1         0         0        0      0     1
##  8  1995                    0      1         0         0        0      0     1
##  9  1995                    0      0         0         0        0      1     0
## 10  1995                    0      0         0         0        0      1     0
## # ... with 15,974 more rows, and 26 more variables: Documentary <dbl>,
## #   Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>, Horror <dbl>, IMAX <dbl>,
## #   Musical <dbl>, Mystery <dbl>, Romance <dbl>, Sci-Fi <dbl>, Thriller <dbl>,
## #   War <dbl>, Western <dbl>, wtitle <chr>, rank <int>, title <chr>,
## #   lifetime_gross <int>, titleId <chr>, isOriginalTitle <dbl>,
## #   averageRating <dbl>, numVotes <dbl>, isAdult <dbl>, runtimeMinutes <chr>,
## #   movieId <int>, tmdbId <int>, avg_rating_grouplense <dbl>
```

```
main_df = merge4 %>% relocate(lifetime_gross) %>%
  select(-c("wtitle","rank","isOriginalTitle","titleId","movieId","tmdbId","title")) %>%  mutate(runtime
main_df
```

```
## # A tibble: 15,984 x 27
##    lifetime_gross  year `(no genres listed)` Action Adventure Animation Children
##             <int> <int>                <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
```

```
##  1      191796233  1995                  0      0         1         1        1
##  2      100475249  1995                  0      0         1         0        1
##  3       71518503  1995                  0      0         0         0        0
##  4       67052156  1995                  0      0         0         0        0
##  5       76594107  1995                  0      0         0         0        0
##  6       67436818  1995                  0      1         0         0        0
##  7       67436818  1995                  0      1         0         0        0
##  8       67436818  1995                  0      1         0         0        0
##  9       53672080  1995                  0      0         0         0        0
## 10       53672080  1995                  0      0         0         0        0
## # ... with 15,974 more rows, and 20 more variables: Comedy <dbl>, Crime <dbl>,
## #   Documentary <dbl>, Drama <dbl>, Fantasy <dbl>, Film-Noir <dbl>,
## #   Horror <dbl>, IMAX <dbl>, Musical <dbl>, Mystery <dbl>, Romance <dbl>,
## #   Sci-Fi <dbl>, Thriller <dbl>, War <dbl>, Western <dbl>,
## #   averageRating <dbl>, numVotes <dbl>, isAdult <dbl>, runtimeMinutes <int>,
## #   avg_rating_grouplense <dbl>
```

```
#rating_grouplense
```

Description of the final data

We have 15,984 observations in the data and 27 features

Each observation represent a movie

lifetime_gross -(continuous) this is our respone variable and its a numeric column that shows the total revenues of each film

year -(continuous) the year the movie was made

(no geners listed) -(categorical) if the movie didnt have any genere listed

columns 4-19 - (categorical) one hot encoding that each one represent a different genere

averageRating - (continuous) the average rating the movie got in IMBd

numVotes - (continuous) the number of votes for the rating

isAdult - (categorical) if the movie is for adults

runtimeMinutes - (continuous) the amount of time in minutes each movie is

avg_rating_grouplense - (continuous) the average rating the movie got in grouplense

EDA

After a very very long process of data tyding and wrangling we will do some exploratory data analysis

lets look at the data

```
describe(main_df)
```

```
##                      vars     n     mean       sd   median  trimmed      mad
## lifetime_gross          1 15984 2.45e+07 5.36e+07 3.68e+06 1.23e+07 5.43e+06
## year                    2 15984 2.00e+03 1.86e+01 2.00e+03 2.00e+03 1.48e+01
## (no genres listed)      3 15984 4.00e-02 1.90e-01 0.00e+00 0.00e+00 0.00e+00
```

```
## Action                    4 15984 1.50e-01 3.50e-01 0.00e+00 6.00e-02 0.00e+00
## Adventure                 5 15984 8.00e-02 2.80e-01 0.00e+00 0.00e+00 0.00e+00
## Animation                 6 15984 3.00e-02 1.70e-01 0.00e+00 0.00e+00 0.00e+00
## Children                  7 15984 6.00e-02 2.30e-01 0.00e+00 0.00e+00 0.00e+00
## Comedy                    8 15984 2.80e-01 4.50e-01 0.00e+00 2.30e-01 0.00e+00
## Crime                     9 15984 1.10e-01 3.10e-01 0.00e+00 1.00e-02 0.00e+00
## Documentary              10 15984 8.00e-02 2.70e-01 0.00e+00 0.00e+00 0.00e+00
## Drama                    11 15984 4.90e-01 5.00e-01 0.00e+00 4.80e-01 0.00e+00
## Fantasy                  12 15984 5.00e-02 2.20e-01 0.00e+00 0.00e+00 0.00e+00
## Film-Noir                13 15984 1.00e-02 8.00e-02 0.00e+00 0.00e+00 0.00e+00
## Horror                   14 15984 9.00e-02 2.80e-01 0.00e+00 0.00e+00 0.00e+00
## IMAX                     15 15984 1.00e-02 9.00e-02 0.00e+00 0.00e+00 0.00e+00
## Musical                  16 15984 2.00e-02 1.40e-01 0.00e+00 0.00e+00 0.00e+00
## Mystery                  17 15984 5.00e-02 2.10e-01 0.00e+00 0.00e+00 0.00e+00
## Romance                  18 15984 1.60e-01 3.60e-01 0.00e+00 7.00e-02 0.00e+00
## Sci-Fi                   19 15984 6.00e-02 2.40e-01 0.00e+00 0.00e+00 0.00e+00
## Thriller                 20 15984 2.00e-01 4.00e-01 0.00e+00 1.20e-01 0.00e+00
## War                      21 15984 3.00e-02 1.60e-01 0.00e+00 0.00e+00 0.00e+00
## Western                  22 15984 1.00e-02 1.10e-01 0.00e+00 0.00e+00 0.00e+00
## averageRating            23 15984 6.26e+00 1.04e+00 6.40e+00 6.32e+00 1.04e+00
## numVotes                 24 15984 4.37e+04 1.15e+05 5.31e+03 1.72e+04 7.29e+03
## isAdult                  25 15984 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
## runtimeMinutes           26 15984 1.03e+02 2.99e+01 9.90e+01 1.01e+02 1.48e+01
## avg_rating_grouplense    27 15584 3.10e+00 6.10e-01 3.17e+00 3.13e+00 5.20e-01
##                           min      max    range  skew kurtosis       se
## lifetime_gross           74.0 7.61e+08 7.61e+08  5.12    39.13 4.24e+05
## year                   1891.0 2.02e+03 1.28e+02 -1.75     3.65 1.50e-01
## (no genres listed)        0.0 1.00e+00 1.00e+00  4.97    22.74 0.00e+00
## Action                    0.0 1.00e+00 1.00e+00  2.00     2.01 0.00e+00
## Adventure                 0.0 1.00e+00 1.00e+00  3.01     7.07 0.00e+00
## Animation                 0.0 1.00e+00 1.00e+00  5.61    29.47 0.00e+00
## Children                  0.0 1.00e+00 1.00e+00  3.88    13.08 0.00e+00
## Comedy                    0.0 1.00e+00 1.00e+00  0.97    -1.06 0.00e+00
## Crime                     0.0 1.00e+00 1.00e+00  2.49     4.23 0.00e+00
## Documentary               0.0 1.00e+00 1.00e+00  3.05     7.29 0.00e+00
## Drama                     0.0 1.00e+00 1.00e+00  0.05    -2.00 0.00e+00
## Fantasy                   0.0 1.00e+00 1.00e+00  3.98    13.86 0.00e+00
## Film-Noir                 0.0 1.00e+00 1.00e+00 12.99   166.86 0.00e+00
## Horror                    0.0 1.00e+00 1.00e+00  2.96     6.75 0.00e+00
## IMAX                      0.0 1.00e+00 1.00e+00 10.66   111.67 0.00e+00
## Musical                   0.0 1.00e+00 1.00e+00  6.75    43.60 0.00e+00
## Mystery                   0.0 1.00e+00 1.00e+00  4.24    15.94 0.00e+00
## Romance                   0.0 1.00e+00 1.00e+00  1.89     1.56 0.00e+00
## Sci-Fi                    0.0 1.00e+00 1.00e+00  3.56    10.68 0.00e+00
## Thriller                  0.0 1.00e+00 1.00e+00  1.52     0.32 0.00e+00
## War                       0.0 1.00e+00 1.00e+00  5.98    33.82 0.00e+00
## Western                   0.0 1.00e+00 1.00e+00  9.06    80.02 0.00e+00
## averageRating             1.6 9.30e+00 7.70e+00 -0.72     0.91 1.00e-02
## numVotes                 11.0 2.46e+06 2.46e+06  6.37    64.49 9.12e+02
## isAdult                   0.0 0.00e+00 0.00e+00   NaN      NaN 0.00e+00
## runtimeMinutes            1.0 8.77e+02 8.76e+02  5.90   104.90 2.40e-01
## avg_rating_grouplense     0.5 5.00e+00 4.50e+00 -0.74     2.09 0.00e+00
```

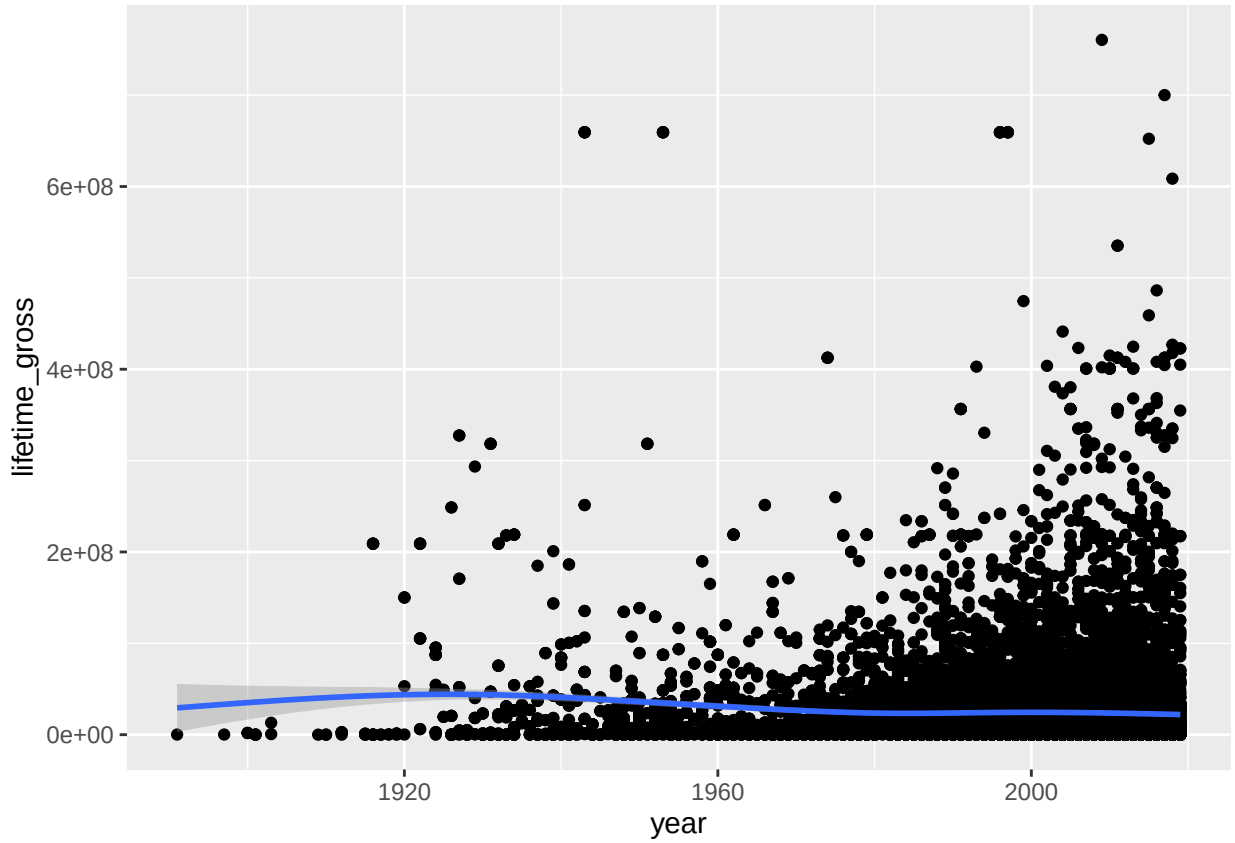
We can see a few intresting things from this. For example we can see that the isadult column

have a mean nedian and sd of 0 meaning the entire column is zero and therefor we should remove it. Another thing is that we can see that the maximum revenue a movie made is 760 million dollars.

```
main_df = main_df%>%select(-isAdult)
```

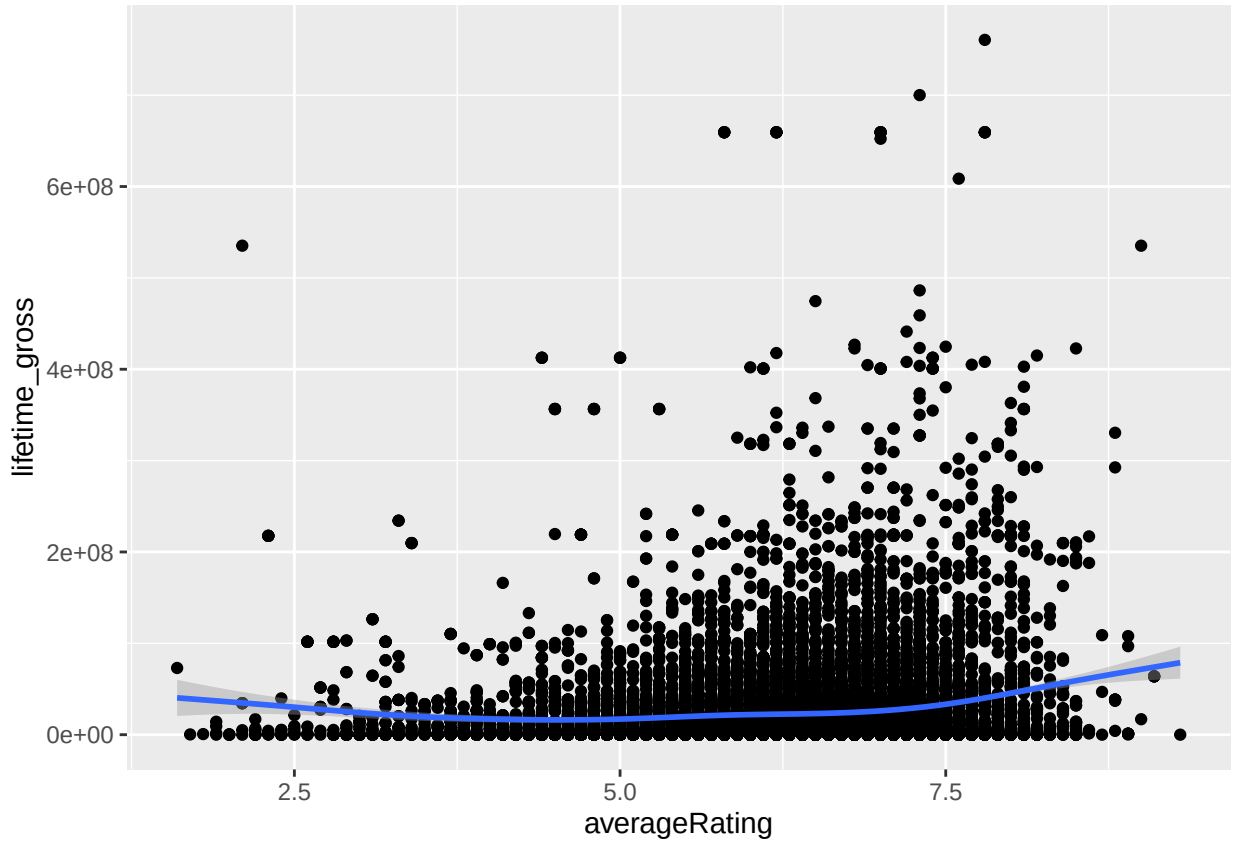
We will plot a few dependencies on lifetime_gross, to get a better overview of our data.

```
main_df %>% ggplot(aes(y = lifetime_gross, x = year)) +
  geom_point() +
  geom_smooth()
```



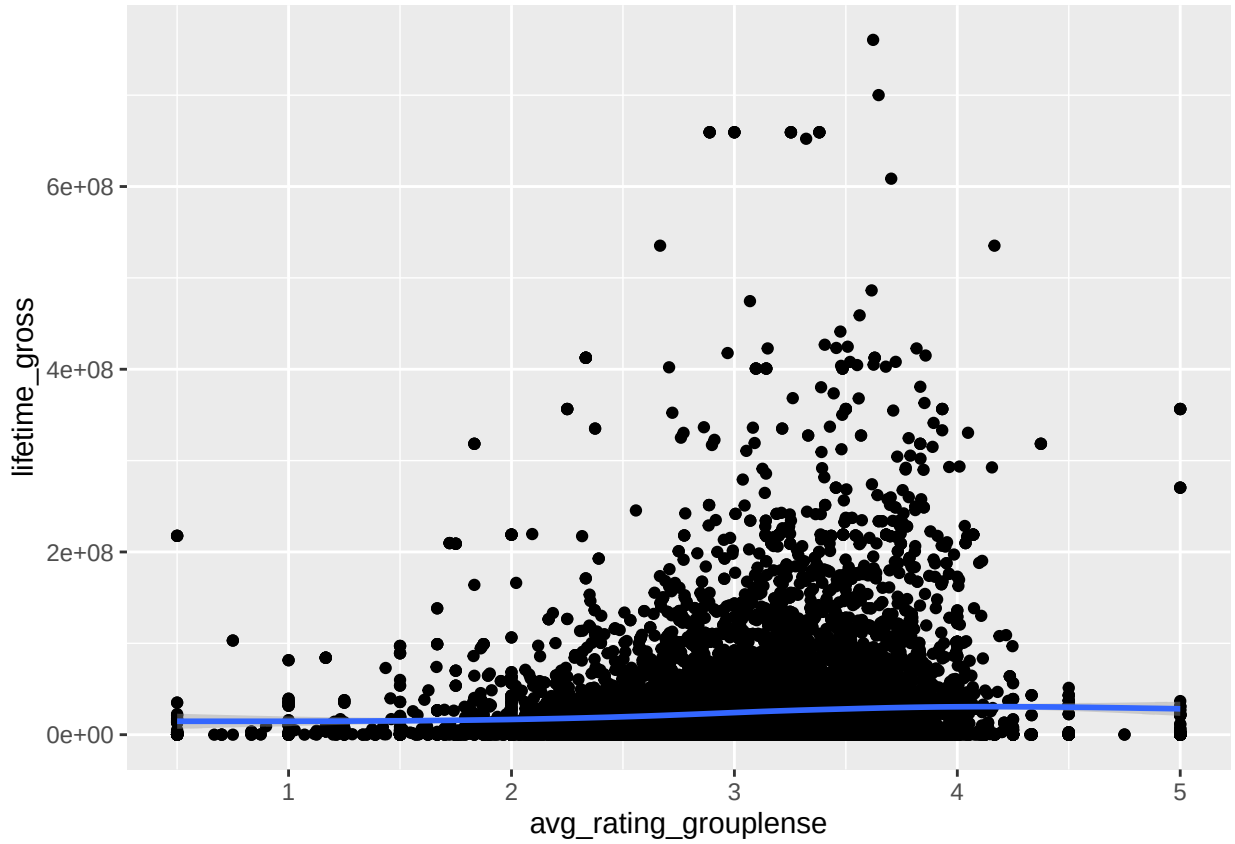
According to our data we can see there is a slight decrease of lifetime_gross over the duration of a movie. This makes sense, older movies in general have more time passed so that you could earn more money over a longer duration.

```
main_df %>% ggplot(aes(y = lifetime_gross, x = averageRating)) +
  geom_point() +
  geom_smooth()
```



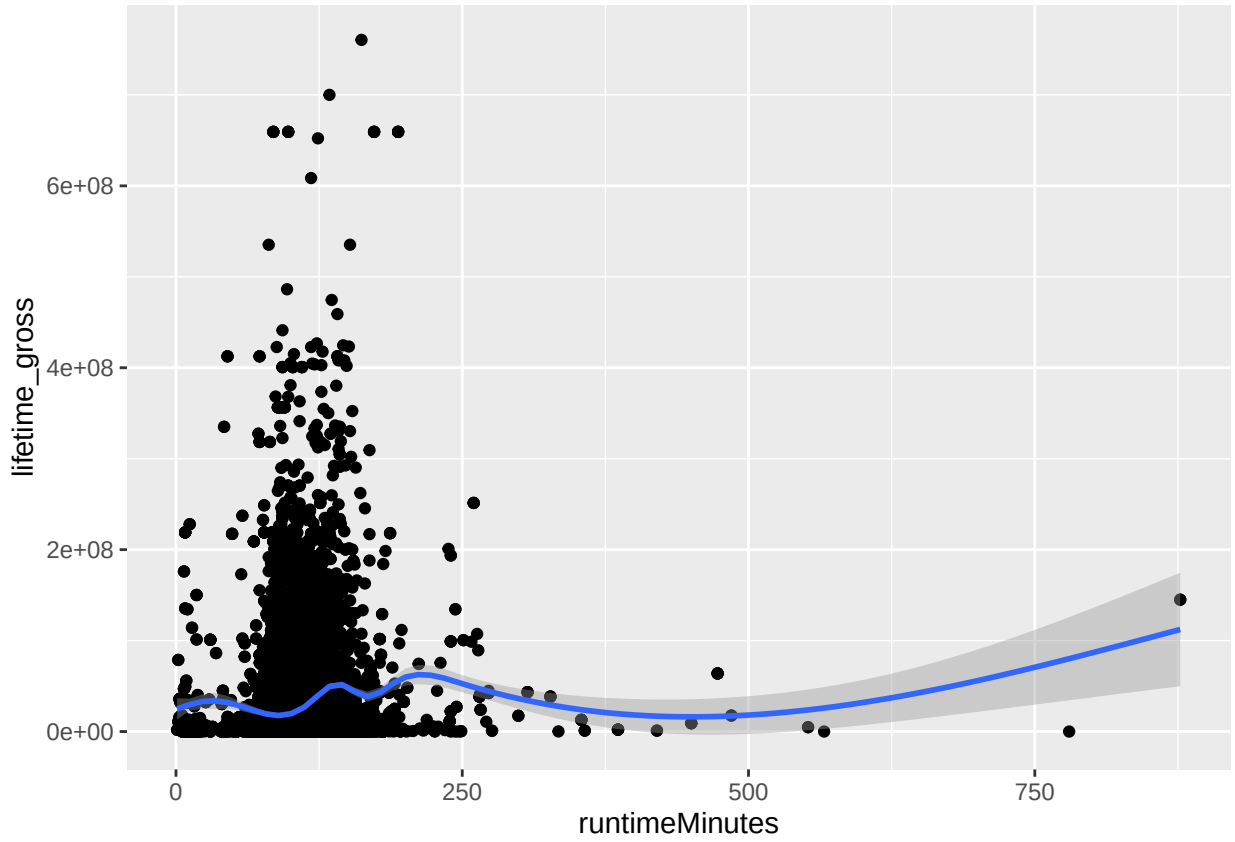
Interesting enough there seems to be a curve if we compare imdb ratings and gross. Easy explainable is the right side where a higher ration correlates with more revenue but interesting enough bad movies also get a lot of revenue. Lets compare that to our second movie rating dataset!

```
main_df %>% ggplot(aes(y = lifetime_gross, x = avg_rating_grouplense)) +
  geom_point() +
  geom_smooth()
```



On the other hand we see that movies with a rating of 4 in gouplense have higher lfetime gross tha other movies. One might explain this because people that rate movies watch alot of movies and thus some concept of the movies a critic might find overused while the casual watcher (here is where the company make the most money) enjoy it more.

```
main_df %>% ggplot(aes(y = lifetime_gross, x = runtimeMinutes)) +
  geom_point() +
  geom_smooth()
```



And lastly it seems like that there is a perfect length of a movie which is around 120 minutes.

Methods and Predictions

Linear regression

Now we shall run some analysis on our data.

```
set.seed(11)
n = nrow(main_df)
train_samples = sample(1:n, round(0.8*n))
```

```
main_df_train = main_df[train_samples,]
main_df_test = main_df[-train_samples,]
```

```
lm_fit = lm(lifetime_gross ~ avg_rating_grouplense, data = main_df_train)

prediction = predict(lm_fit,  newdata = main_df_test)

tibble("Predicted gross" = prediction,
       "Actual gross" = main_df_test$lifetime_gross) %>%
  head(10)
```

```
## # A tibble: 10 x 2
##    `Predicted gross` `Actual gross`
##                <dbl>          <int>
##  1         25166459.       71518503
##  2         24992295.       23920048
##  3         26381710.       11348324
##  4         29492375.       42512375
##  5         32726452.       43182776
##  6         26704231.        4257354
##  7         23439680.       35431113
##  8         25189706.       30303072
##  9         25130512.        2844379
## 10         26828491.       27112329
```

```
summary(lm_fit)
```

```
##
## Call:
## lm(formula = lifetime_gross ~ avg_rating_grouplense, data = main_df_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -36956217 -24788944 -18727999   2401259 732293688
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            5227228    2516327    2.08    0.038 *
## avg_rating_grouplense  6345975     797919    7.95    2e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.4e+07 on 12453 degrees of freedom
##   (332 observations deleted due to missingness)
## Multiple R-squared:  0.00505,    Adjusted R-squared:  0.00497
## F-statistic: 63.3 on 1 and 12453 DF,  p-value: 1.97e-15
```

```r
lm_fit = lm(lifetime_gross ~ averageRating, data = main_df_train)

prediction = predict(lm_fit,  newdata = main_df_test)

tibble("Predicted gross" = prediction,
       "Actual gross" = main_df_test$lifetime_gross) %>%
  head(10)
```

```
## # A tibble: 10 x 2
##    `Predicted gross` `Actual gross`
##                <dbl>          <int>
##  1         26709618.       71518503
##  2         20875912.       23920048
##  3         28654187.       11348324
##  4         34001750.       42512375
##  5         27195760.       43182776
##  6         27195760.        4257354
##  7         21848197.       35431113
##  8         24765050.       30303072
##  9         28654187.        2844379
## 10         27195760.       27112329
```

```
summary(lm_fit)
```

```
##
## Call:
## lm(formula = lifetime_gross ~ averageRating, data = main_df_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -39324045 -24697319 -17876113   2411808 728450443
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -5861906    2871844   -2.04    0.041 *
## averageRating  4861421     453100   10.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 53400000 on 12785 degrees of freedom
## Multiple R-squared:  0.00892,    Adjusted R-squared:  0.00885
## F-statistic:  115 on 1 and 12785 DF,  p-value: <2e-16
```

we can see that a lm model is not very suitable for our data, this is because we are only looking at one variable so lets expand it and to have a linear regression for multiple variables

```
lm_fit = lm(lifetime_gross ~., data = main_df_train)
summary(lm_fit)
```

```
##
## Call:
## lm(formula = lifetime_gross ~ ., data = main_df_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -4.11e+08 -1.37e+07 -5.87e+06  2.22e+06  6.48e+08
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             4.20e+08   4.48e+07    9.37  < 2e-16 ***
## year                   -2.03e+05   2.23e+04   -9.10  < 2e-16 ***
## `(no genres listed)`    2.34e+06   2.41e+06    0.97  0.33066
## Action                  1.52e+07   1.27e+06   11.98  < 2e-16 ***
## Adventure               1.79e+07   1.60e+06   11.21  < 2e-16 ***
## Animation               2.69e+07   2.62e+06   10.27  < 2e-16 ***
## Children                1.34e+07   2.04e+06    6.60  4.3e-11 ***
## Comedy                  2.13e+06   1.04e+06    2.05  0.04023 *
## Crime                  -2.95e+06   1.35e+06   -2.18  0.02937 *
## Documentary            -4.27e+06   1.79e+06   -2.39  0.01706 *
## Drama                  -3.51e+06   1.02e+06   -3.44  0.00057 ***
## Fantasy                 1.69e+07   1.87e+06    9.01  < 2e-16 ***
## `Film-Noir`            -1.30e+07   5.34e+06   -2.43  0.01508 *
## Horror                 -8.38e+04   1.55e+06   -0.05  0.95699
## IMAX                    3.19e+07   4.48e+06    7.11  1.2e-12 ***
## Musical                 1.70e+06   2.84e+06    0.60  0.54987
## Mystery                 3.62e+06   1.89e+06    1.91  0.05564 .
## Romance                 4.82e+06   1.13e+06    4.25  2.1e-05 ***
## `Sci-Fi`               -1.43e+06   1.69e+06   -0.84  0.39829
## Thriller               -8.24e+05   1.17e+06   -0.71  0.47989
## War                    -4.63e+06   2.53e+06   -1.83  0.06678 .
## Western                -4.87e+06   3.69e+06   -1.32  0.18693
## averageRating          -1.73e+06   5.37e+05   -3.23  0.00125 **
## numVotes                2.19e+02   3.84e+00   57.06  < 2e-16 ***
## runtimeMinutes          4.91e+04   1.34e+04    3.66  0.00025 ***
## avg_rating_grouplense   4.85e+05   8.61e+05    0.56  0.57352
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43700000 on 12429 degrees of freedom
##   (332 observations deleted due to missingness)
## Multiple R-squared:  0.349,  Adjusted R-squared:  0.348
## F-statistic:  267 on 25 and 12429 DF,  p-value: <2e-16
```

```
rmse(main_df_test$lifetime_gross,prediction)
```

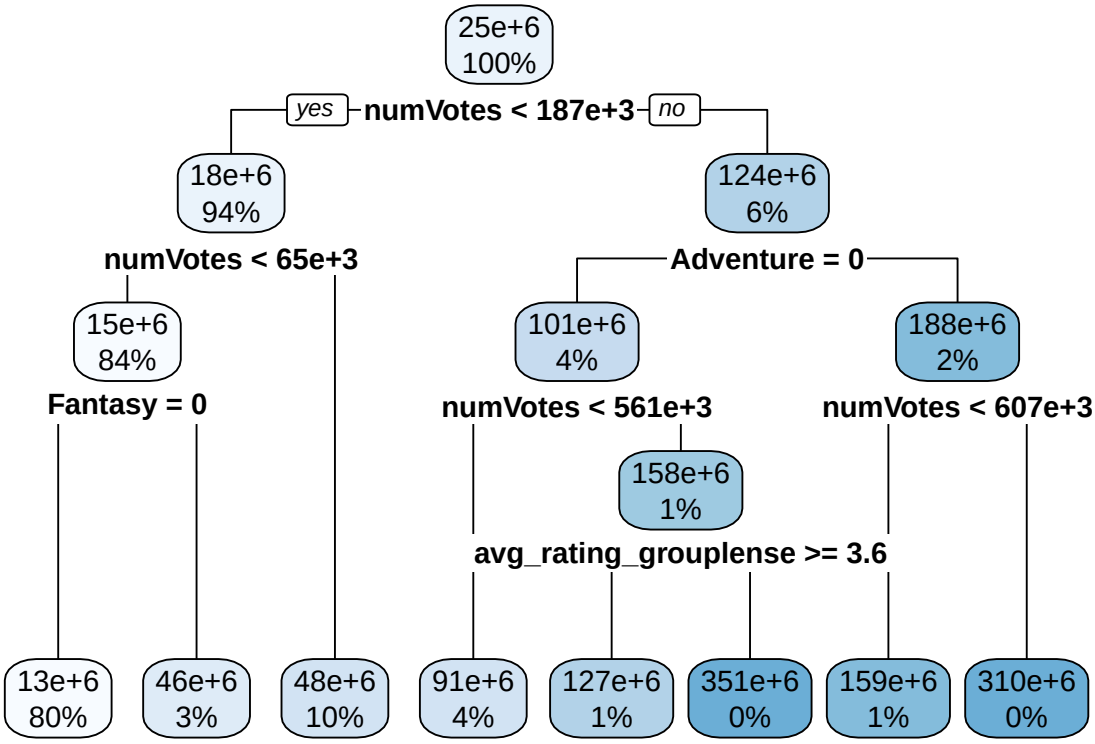
```
## [1] 5.3e+07
```

Now we can see that we got much better results.Our adjusted r-squared increased to 0.348. Further more we can see a few interesting results. year seems to have a negative impact on

the revenue as we already saw in our EDA section. Different generes seems to have different impact on the revenue like animation and action have a positive impact where drama and documentary have a negative one. This is to be expected since we know mosr blockbusters are action and animation where drama and documentary are less of a money grab.

Decision Trees

```
deep_tree_fit = rpart(lifetime_gross ~ .,
                          method = "anova",
                          parms = list(split = "gini"),
                          control = rpart.control(minsplit = 5),
                          data = main_df_train)
rpart.plot(deep_tree_fit)
```




```
deep_tree_fit
```

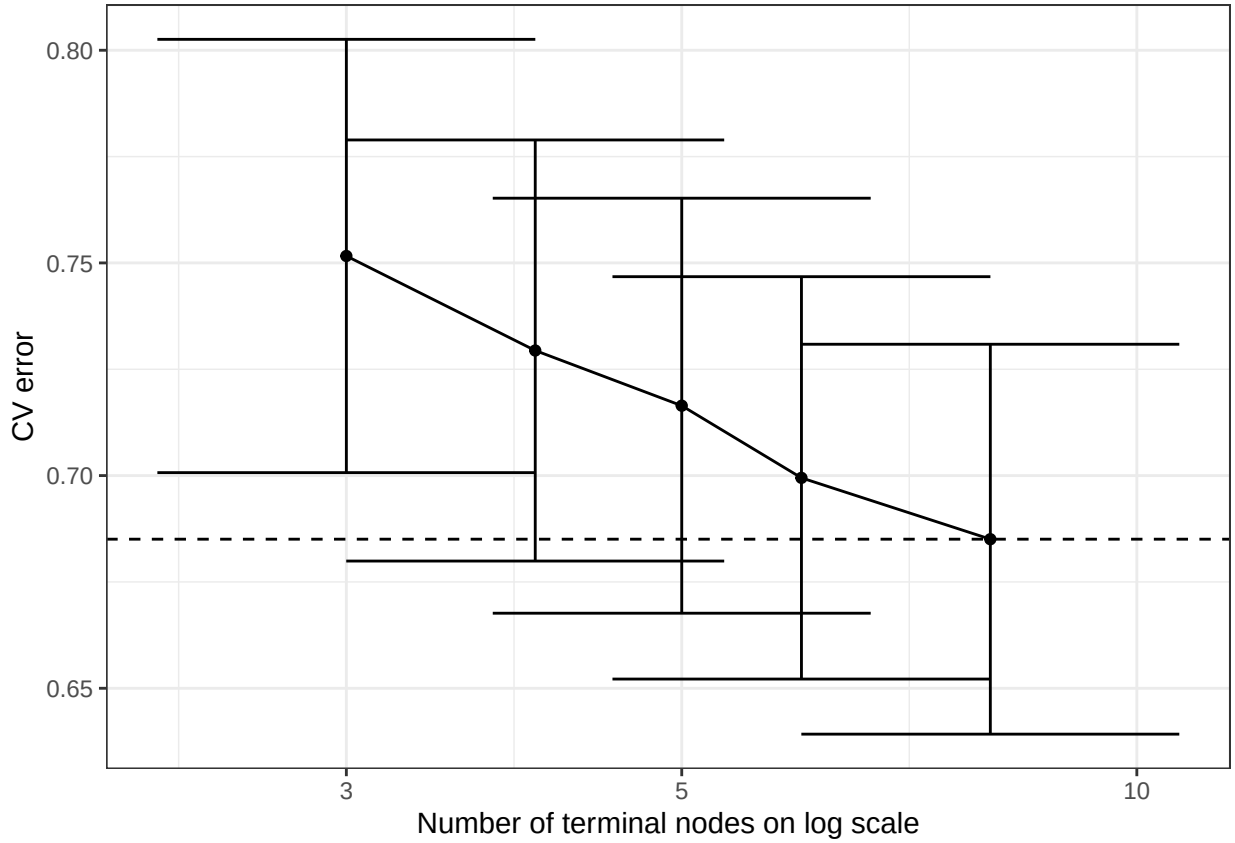
```
## n= 12787
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 12787 3.68e+19 2.45e+07
##    2) numVotes< 1.87e+05 12025 2.01e+19 1.82e+07
##      4) numVotes< 6.52e+04 10700 1.63e+19 1.46e+07
##        8) Fantasy< 0.5 10255 1.35e+19 1.32e+07 *
##        9) Fantasy>=0.5 445 2.34e+18 4.61e+07 *
##      5) numVotes>=6.52e+04 1325 2.50e+18 4.79e+07 *
##    3) numVotes>=1.87e+05 762 8.73e+18 1.24e+08
##      6) Adventure< 0.5 563 4.36e+18 1.01e+08
```

```
##       12) numVotes< 5.61e+05 476 2.26e+18 9.06e+07 *
##       13) numVotes>=5.61e+05 87 1.77e+18 1.58e+08
##         26) avg_rating_grouplense>=3.63 75 9.37e+17 1.27e+08 *
##         27) avg_rating_grouplense< 3.63 12 3.16e+17 3.51e+08 *
##      7) Adventure>=0.5 199 3.25e+18 1.88e+08
##       14) numVotes< 6.07e+05 160 1.64e+18 1.59e+08 *
##       15) numVotes>=6.07e+05 39 9.01e+17 3.10e+08 *
```

We can see that number of votes seems to be the leading factor in deciding the splits splits This is a non prune tree so lets get a better tree and see the results

```
cp_table = deep_tree_fit$cptable %>% as_tibble()

cp_table %>%
  filter(nsplit >= 2) %>%
  ggplot(aes(x = nsplit+1, y = xerror,
             ymin = xerror - xstd, ymax = xerror + xstd)) +
  scale_x_log10() +
  geom_point() + geom_line() +
  geom_errorbar(width = 0.25) +
  xlab("Number of terminal nodes on log scale") + ylab("CV error") +
  geom_hline(aes(yintercept = min(xerror)), linetype = "dashed") +
  theme_bw()
```

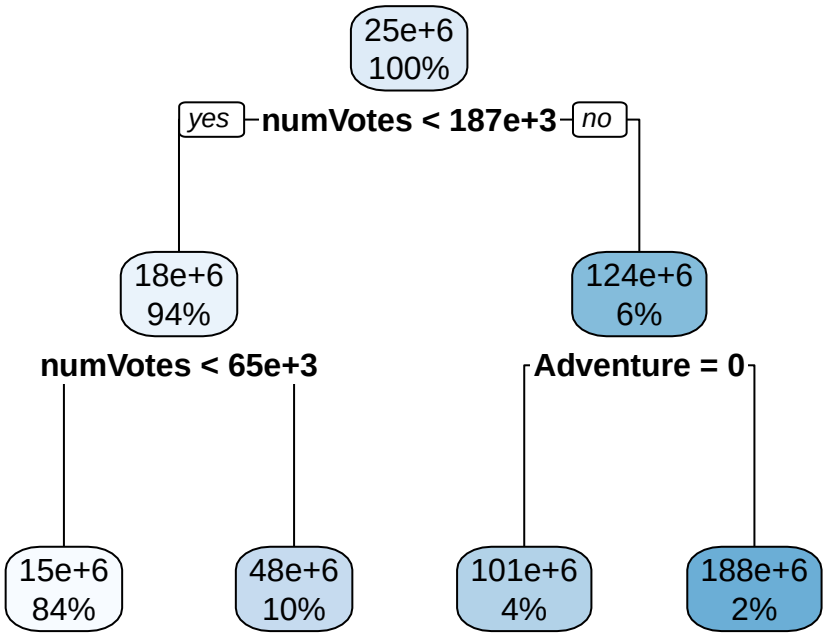


We can see from the plot above that the error keep decreasing which indicate that we do need a longer tree, the problem is that we don't really have a lot of observatons to do it becuase from tree looks most of the nods stoped due to lack f observation in them and so even if we didn't get overfitt yet we should just continue like thi.

```
optimal_tree_info = cp_table %>%
  filter(xerror - xstd < min(xerror)) %>%
  arrange(nsplit) %>% head(1)
optimal_tree_info$nsplit
```

```
## [1] 3
```

```
optimal_tree = prune(tree = deep_tree_fit, cp = optimal_tree_info$CP)
rpart.plot(optimal_tree)
```




```
prediction = predict(optimal_tree, main_df_test)

rmse(prediction, main_df_test$lifetime_gross) #52072302
```

```
## [1] 42419984
```

We see the pruned tree is a much smaller. Pruning successfull, the RMSE althogh its lower than the one we got in the linear regression its still quite high. We shall now use randomForest to reduce variance in our data. This will hopefully improve our predictive model.

One problem with randomForest is that the names of the columns are not allowed to contain special characters else the randomForest command will not compile, thus we will mutate our column names before we run the model function,

```
a = main_df_train %>% mutate(nogenre = `(no genres listed)`, filmnoir = `Film-Noir`, scifi = `Sci-Fi`) %

rf_fit = randomForest(lifetime_gross ~ ., data = a, mtry = 3,
                          importance = TRUE, na.action = na.omit)
num_features = ncol(a) - 1
mtry = floor(sqrt(num_features))
mtry
```

```
## [1] 5
```

```
rf_fit
```

```
##
## Call:
##  randomForest(formula = lifetime_gross ~ ., data = a, mtry = 3,      importance = TRUE, na.action = n
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 1.56e+15
##                     % Var explained: 46.7
```

Conclusion

In conclusion we can see that our tested metrics do have a correlation with lifetime gross of a movie, we could also see that for example a few metrics are more influential then other. One such example we have showed in a broader way in our regression models where we compared the two ratings - imdb and grouplense. Here we could see that the imdb metric has a higher (linear) correlation with our response. The best predictive methods are ensamble methods. All of these models come to a conclusion that we can predict the lifetime gross of a movie just by analysing meta data and without watching the movie itself.