Final Group Project - Movie Analysis

Group 7

2024-04-28

Data Origin

1. **MovieLens-GroupLens Research**

   Most of our current data is from the small version of the MovieLens dataset, collected and provided by GroupLens Research. The dataset includes several files, such as `movies.csv`, `ratings.csv`, and `links.csv`, which contain information about movies, user ratings, and external links to other movie databases. The MovieLens dataset is regularly updated and comes in different sizes, including the small version with 100,000 ratings applied to 9,000 movies by 600 users. The datasets are available for download from the GroupLens website.

2. **IMDb**

   We wrote a Python crawler to scrape cast data from IMDb, obtaining `cast.csv`. IMDb is a comprehensive movie database that provides information about movies, TV shows, and celebrities. Our crawler extracted cast details for each movie, allowing us to enrich our dataset with additional information about the actors and actresses involved in the films. This data complements the MovieLens dataset, providing a more complete picture of the movies and their cast members.

Research Questions

1. Does the variety of content, as measured by the number of genres, influence the overall rating of movies?
2. Which genres have been more popular among viewers in the past three decades?
3. Which genres have generated more revenue and interest compared to others?
4. Does the popularity of top cast members necessarily lead to a larger consumer base and business success for movies?
5. Does the frequent casting of popular actors in high-quality films necessarily result in higher audience preference and improved ratings for those movies?

Import Libraries and Data

```
library(dplyr)
library(tidyverse)
library(ggplot2)
library(caret)
library(stringr)
library(gbm)
set.seed(123)
# Read data
links <- read.csv("data/links.csv")
movies <- read.csv("data/movies.csv")
ratings <- read.csv("data/ratings.csv")
cast <- read.csv("data/cast.csv")
boxOffice <- read.csv("data/boxOffice.csv")
```

```r
# Data Preprocessing
merged_data <- merge(x = links, y = movies, by = "movieId")
ratings <- ratings %>%
  group_by(movieId) %>%
  mutate(avg_rating = mean(rating))
merged_data <- merge(x = ratings, y = merged_data, by = "movieId")
merged_data <- inner_join(merged_data, boxOffice, by="imdbId")
merged_data <- merged_data %>% select(-c("Domestic", "International"))
merged_data <- merged_data %>%
  filter(!is.na(WorldWide))

# Add Genre Count
merged_data$genre_count <- str_count(merged_data$genres, "\\|") + 1

# Merge Cast
merged_data <- merge(cast, merged_data, by = "imdbId")

# Create a function to count occurrences of each actor
get_actor_frequency <- function(df) {
  # Combine all actor columns and create a vector
  actor_vector <- unlist(df[, grepl("Actor", names(df))])
  # Count occurrences of each actor
  actor_frequency <- table(actor_vector)
  return(actor_frequency)
}

# Apply the function to your merged cast data
actor_frequency <- get_actor_frequency(merged_data)
# Calculate actor_score, ignoring NAs
merged_data$actor_score <- apply(merged_data[, grepl("Actor", names(merged_data))], 1, function(row_acto
  # Remove NAs from the current row's actors
  valid_actors <- row_actors[!is.na(row_actors)]
  # Sum the frequencies of valid actors, defaulting to 0 if no valid actors
  sum(actor_frequency[valid_actors], na.rm = TRUE)
})

merged_data$avg_actor_score <- apply(merged_data[, grepl("Actor", names(merged_data))], 1, function(x) {
  # Filter out NULL or missing actor names and then look up their scores
  valid_actors <- x[!is.na(x) & x != ""]
  actor_scores <- actor_frequency[valid_actors]

  # Check if actor_scores is not empty and contains non-NA values
  if (!is.null(actor_scores) && length(actor_scores) > 0 && !any(is.na(actor_scores)) && sum(actor_score
    return(mean(actor_scores[actor_scores > 0], na.rm = TRUE))  # Calculate mean, ignoring NA values
  } else {
    return(0)  # return 0 if no valid actor scores
  }
})

good_movies <- merged_data[merged_data$avg_rating > 3.5, ]

# Count actor occurrences only in good movies
get_good_actor_frequency <- function(df) {
```

```
  actor_vector <- unlist(df[, grepl("Actor", names(df))])
  actor_frequency <- table(actor_vector)
  return(actor_frequency)
}

# Apply the function to the filtered good movies data
good_actor_frequency <- get_good_actor_frequency(good_movies)

# Calculate the sum actor score for each movie based on good occurrences
#merged_data$sum_good_actor_score <- apply(merged_data[, grepl("Actor", names(merged_data))], 1, functi
# Initialize the sum_good_actor_score column with zeros
merged_data$sum_good_actor_score <- numeric(nrow(merged_data))

# Iterate over each row to calculate the sum_good_actor_score
for (i in seq_len(nrow(merged_data))) {
  # If the movie's rating is greater than 3.5, calculate the score
  if (merged_data$avg_rating[i] > 3.5) {
    # Extract actor names as a vector, not as a list
    actor_names <- unlist(merged_data[i, grepl("Actor", names(merged_data))])
    # Sum the frequencies of these actors from the good_actor_frequency table, ignoring NA values
    merged_data$sum_good_actor_score[i] <- sum(good_actor_frequency[actor_names], na.rm = TRUE)
  }
  # If the condition is not met, the score remains 0 as initialized
}




# Calculate the average actor score for each movie based on good occurrences
merged_data$avg_good_actor_score <- apply(merged_data[, grepl("Actor", names(merged_data))], 1, functio
  # Filter out NULL or missing actor names and then look up their scores
  valid_actors <- x[!is.na(x) & x != ""]
  actor_scores <- good_actor_frequency[valid_actors]

  # Check if actor_scores is not empty and contains non-NA values
  if (!is.null(actor_scores) && length(actor_scores) > 0 && !any(is.na(actor_scores)) && sum(actor_scor
    return(mean(actor_scores[actor_scores > 0], na.rm = TRUE))  # Calculate mean, ignoring NA values
  } else {
    return(0)  # return 0 if no valid actor scores
  }
})

# Extracting the year from the title and converting it to a decade
merged_data$title_decade <- gsub(".*\\((\\d{4})\\).*", "\\1", merged_data$title)
merged_data$title_decade <- as.numeric(merged_data$title_decade)
merged_data$decade <- (merged_data$title_decade %/% 10) * 10

# Remove duplicate
merged_data <- merged_data %>% distinct(imdbId, .keep_all = TRUE)

# Split data into training and testing sets
train_indices <- sample(seq_len(nrow(merged_data)), size = 0.7 * nrow(merged_data))
train_data <- merged_data[train_indices, ]
```

```
test_data <- merged_data[-train_indices, ]
```

Model Fitting (Linear Regression) - Genre Count

To fit our model, we first count the number of genres each movie belongs to and create a new column called `genre_count`. Next, we calculate the average rating for each movie and store it in a column named `avg_rating`. Using `genre_count` as our independent variable (x) and `avg_rating` as our dependent variable (y), we then fit a linear model to explore the relationship between the number of genres a movie belongs to and its average rating.

The linear regression analysis exploring the relationship between the number of genres a movie belongs to (`genre_count`) and its average rating (`avg_rating`) shows statistically significant coefficients, with a p-value for `genre_count` less than 2.2e-16. However, the model's predictive power is minimal, as evidenced by a low R^2 of 0.004222, indicating that only about 0.42% of the variance in movie ratings is explained by genre count. The Root Mean Squared Error (RMSE) of approximately 0.564 on testing data suggests moderate prediction errors. This analysis suggests that while the relationship between genre count and average ratings is statistically significant, it is not practically significant, implying that additional variables likely play a more substantial role in determining movie ratings.

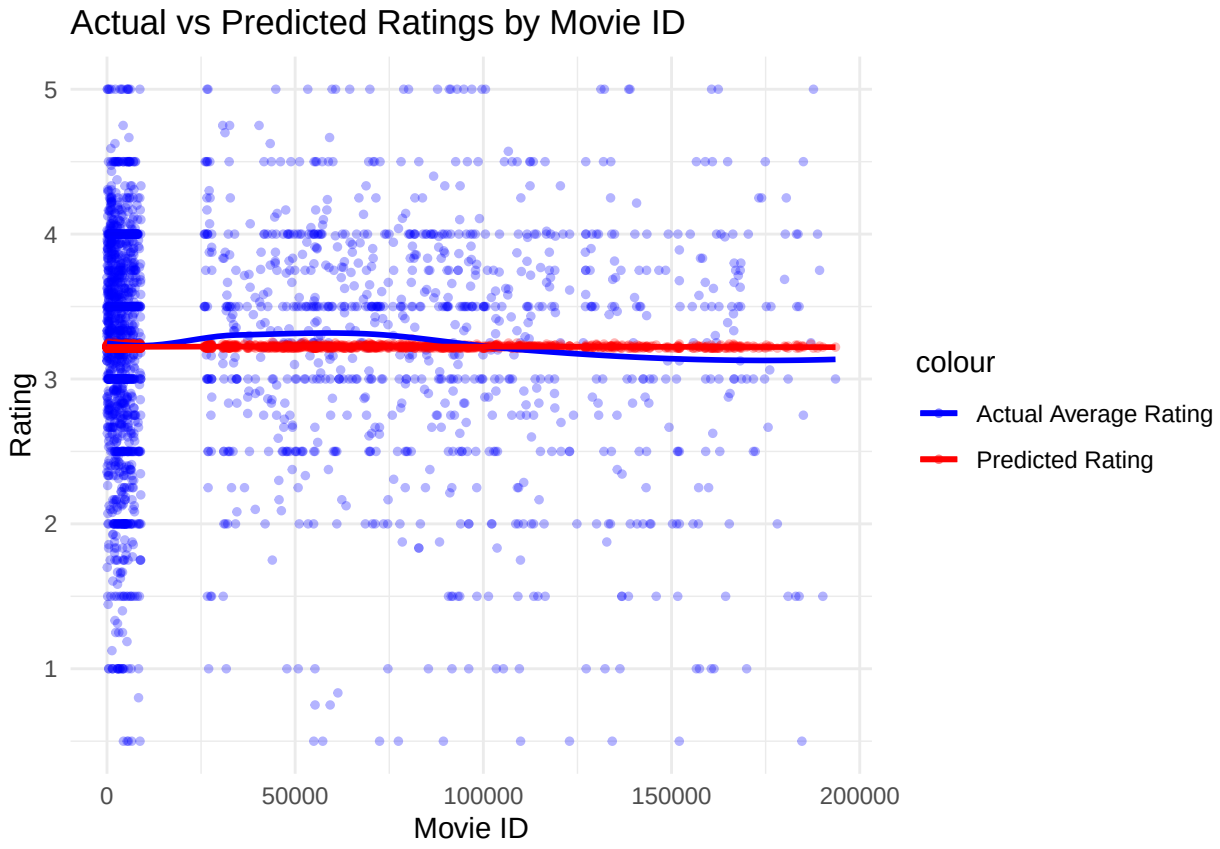
```
# Fit the linear model on training data
model.lm <- lm(avg_rating ~ genre_count, data = train_data)

# Summary of the model to get R^2 and p-values
summary(model.lm)
```

```
##
## Call:
## lm(formula = avg_rating ~ genre_count, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7413 -0.4700  0.1164  0.5896  1.7871
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.205813   0.025170 127.366   <2e-16 ***
## genre_count 0.007097   0.009824   0.722     0.47
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8181 on 5464 degrees of freedom
## Multiple R-squared:  9.55e-05,   Adjusted R-squared:  -8.75e-05
## F-statistic: 0.5218 on 1 and 5464 DF,  p-value: 0.4701
```

```
# Predict on testing data
test_data_lm <- test_data
test_data_lm$predicted_rating <- predict(model.lm, newdata = test_data_lm)

# Evaluate the model using RMSE
mse <- mean((test_data_lm$predicted_rating - test_data_lm$avg_rating)^2)
rmse <- sqrt(mse)
print(paste("Root Mean Squared Error: ", rmse))
```

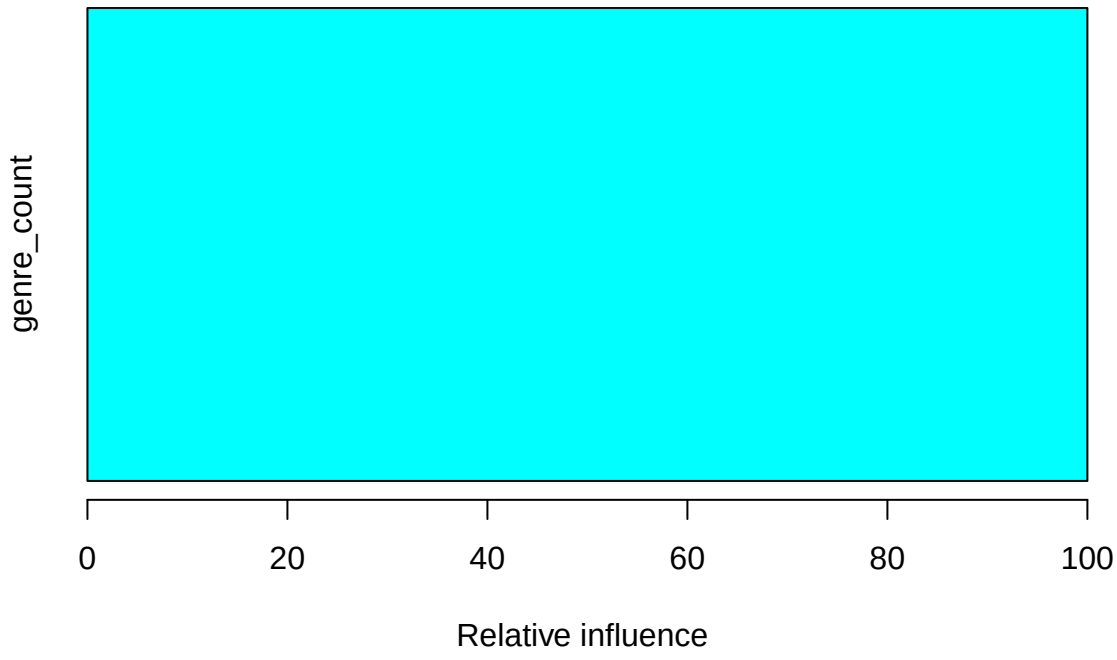
```
## [1] "Root Mean Squared Error:  0.824788144348624"
```

```r
# Plot the ground truth vs predicted values
ggplot(test_data_lm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating",
       title = "Actual vs Predicted Ratings by Movie ID") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



Model Fitting (Gradient Boosting Machine) - Genre Count

```r
# Fit the linear model on training data
model.gbm <- gbm(avg_rating ~ genre_count, data = train_data, distribution = "gaussian", n.trees = 100,

# Summary of the model to get R^2 and p-values
summary(model.gbm)
```

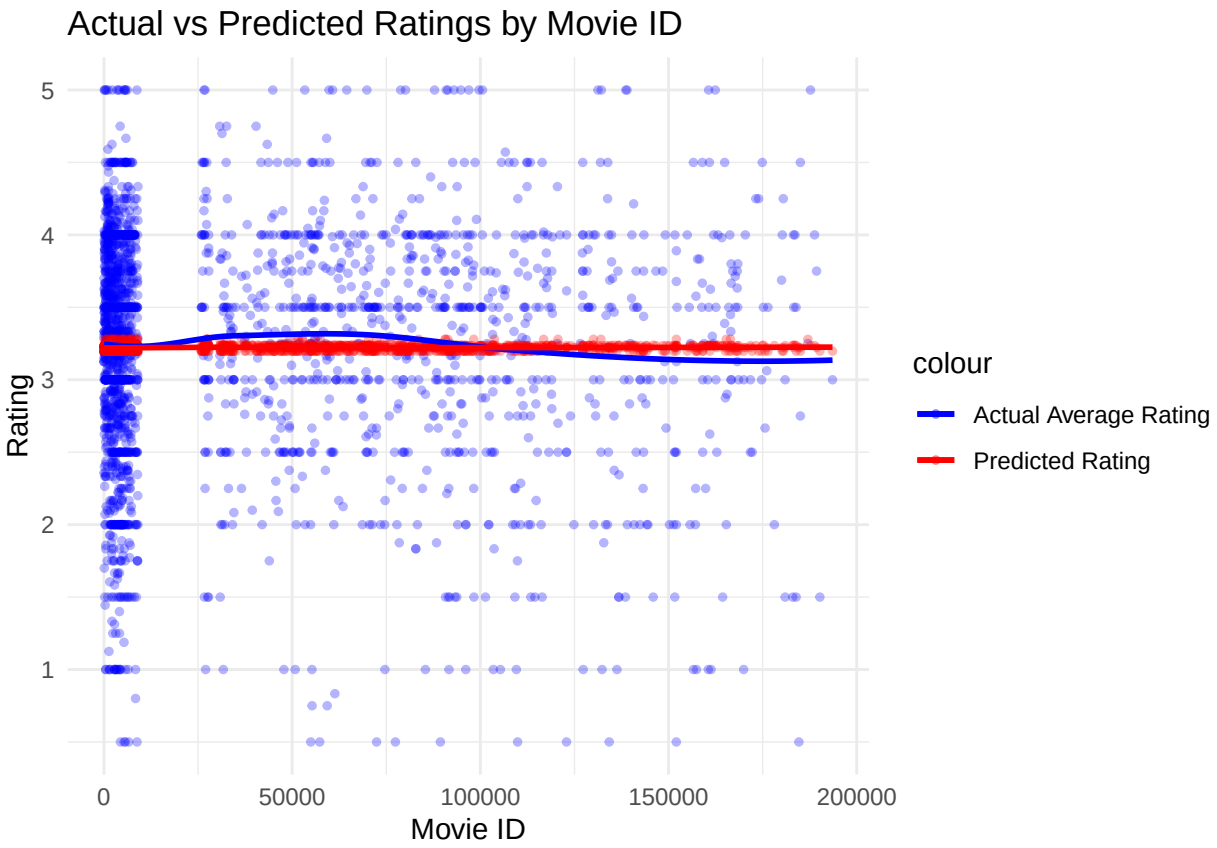



```
##                     var rel.inf
## genre_count genre_count     100
```

```
# Predict on testing data
test_data_gbm <- test_data
test_data_gbm$predicted_rating <- predict(model.gbm, newdata = test_data_gbm)

# Evaluate the model using RMSE
mse <- mean((test_data_gbm$predicted_rating - test_data_gbm$avg_rating)^2)
rmse <- sqrt(mse)
print(paste("Root Mean Squared Error: ", rmse))
```

```
## [1] "Root Mean Squared Error:  0.824388758607909"
```

```
# Plot the ground truth vs predicted values
ggplot(test_data_gbm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating",
       title = "Actual vs Predicted Ratings by Movie ID") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



Model Fitting (Linear Regression) - Genre Count + Average Actor Occurrences

In our second attempt, we introduced an actor score to our model, in addition to genre counts. For each actor, we calculated their actor score based on the number of times they appeared in all the movies. For each movie, the actor scores of all its actors were summed to obtain the movie's `actor_score`. We used `genre_count` and `actor_score` as our independent variables (x) and `avg_rating` as our dependent variable (y). A linear model was then fitted to explore the relationship between these variables and the average rating of the movies.

```
# Fit model on data with both genre count and actor score
model.lm2 <- lm(avg_rating ~ genre_count + avg_actor_score, data = train_data)
summary(model.lm2)
```

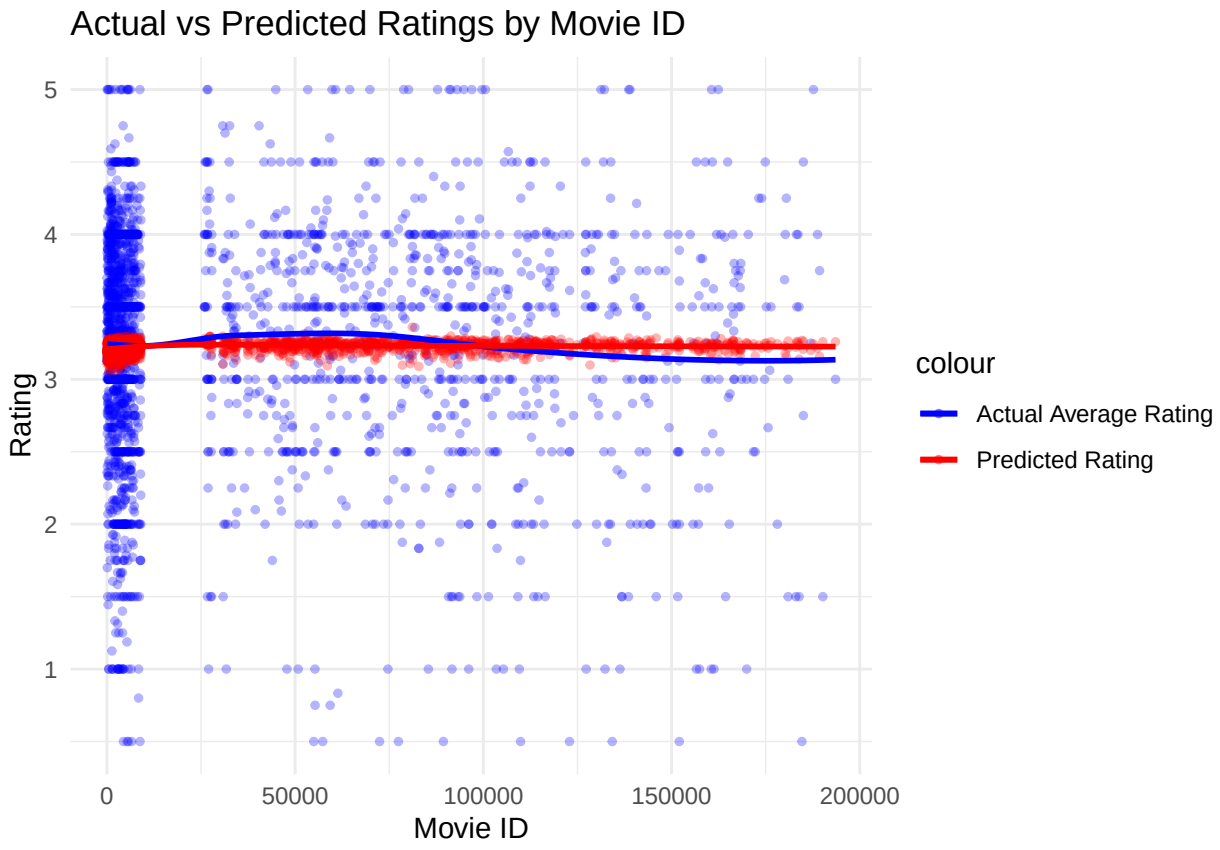
```
##
## Call:
## lm(formula = avg_rating ~ genre_count + avg_actor_score, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7854 -0.4549  0.1230  0.5874  1.8113
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.234e+00  2.650e-02 122.017  < 2e-16 ***
## genre_count       1.303e-02  9.973e-03   1.307 0.191329
## avg_actor_score -3.121e-04  9.287e-05  -3.360 0.000784 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8173 on 5463 degrees of freedom
## Multiple R-squared:  0.002158,   Adjusted R-squared:  0.001793
## F-statistic: 5.907 on 2 and 5463 DF,  p-value: 0.002738
```

```r
test_data_lm2 <- test_data
test_data_lm2$predicted_rating <- predict(model.lm2, newdata = test_data_lm2)
# Evaluate the model using RMSE
mse <- mean((test_data_lm2$predicted_rating - test_data_lm2$avg_rating)^2)
rmse <- sqrt(mse)
print(paste("Root Mean Squared Error: ", rmse))
```

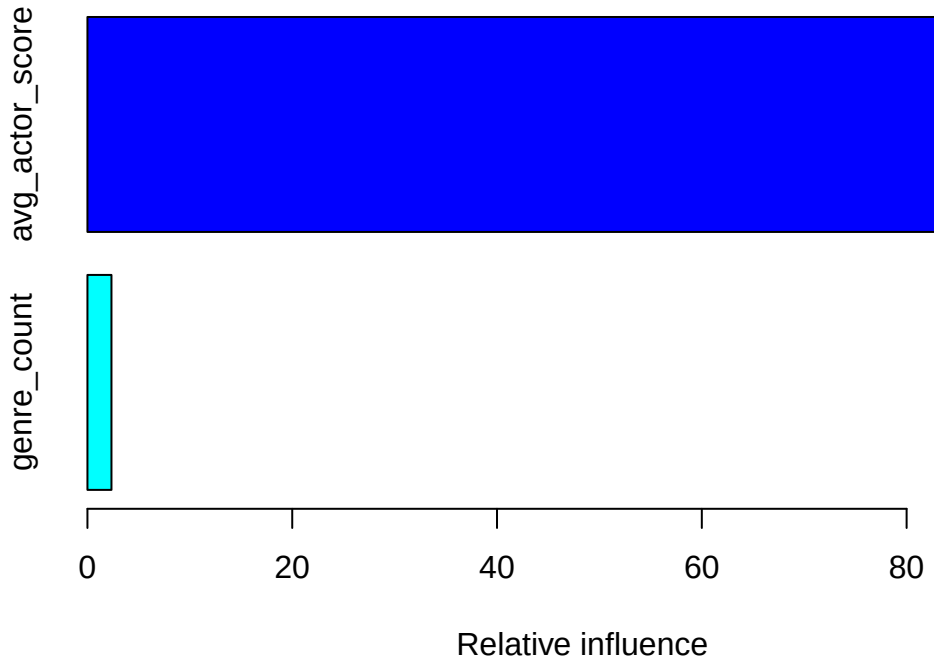
```
## [1] "Root Mean Squared Error:  0.823518719080436"
```

```r
# Plot the ground truth vs predicted values
ggplot(test_data_lm2, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating",
       title = "Actual vs Predicted Ratings by Movie ID") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



Model Fitting (Gradient Boosting Machine) - Genre Count + Average Actor Occurrences

```
# Fit model on data with both genre count and actor score
model.gbm2 <- gbm(avg_rating ~ genre_count + avg_actor_score, data = train_data, distribution = "gaussi
summary(model.gbm2)
```

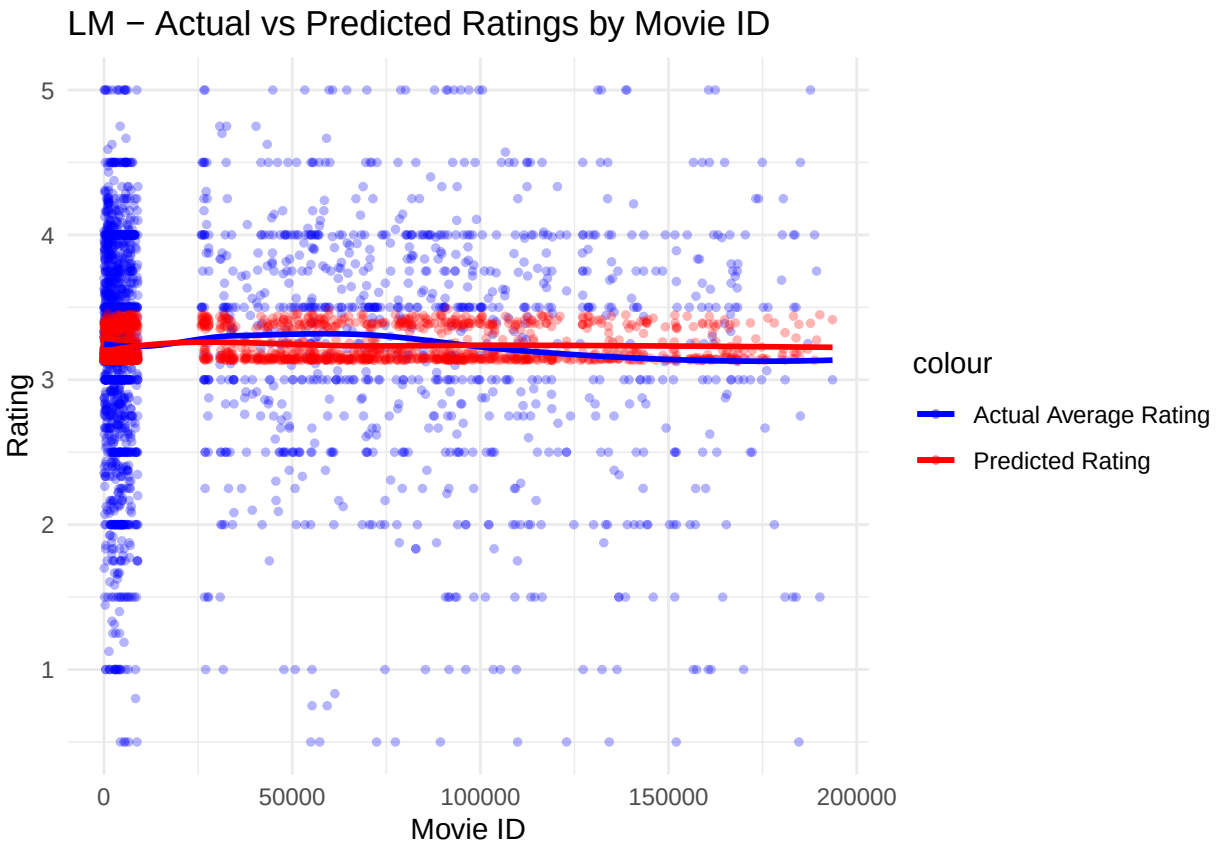



```
##                              var  rel.inf
## avg_actor_score avg_actor_score 97.65274
## genre_count         genre_count  2.34726
```

```
test_data_gbm2 <- test_data
test_data_gbm2$predicted_rating <- predict(model.gbm2, newdata = test_data_gbm2)
# Evaluate the model using RMSE
mse <- mean((test_data_gbm2$predicted_rating - test_data_gbm2$avg_rating)^2)
rmse <- sqrt(mse)
print(paste("Root Mean Squared Error: ", rmse))
```

```
## [1] "Root Mean Squared Error:  0.807833068713011"
```

```
# Plot the ground truth vs predicted values
ggplot(test_data_gbm2, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating",
       title = "LM - Actual vs Predicted Ratings by Movie ID") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



Model Fitting (Linear Regression) - Genre Count + Good Actor Occurrences

In our third attempt, we refined the actor score from our second attempt by considering only the number of appearances in movies with a rating higher than 3.5. We defined these as good movies. For each actor, we calculated their actor score based on their appearances in good movies. For each movie, we summed the actor scores of all its actors to obtain the movie's `good_actor_score`. We used `genre_count` and `good_actor_score` as our independent variables (x) and `avg_rating` as our dependent variable (y). A linear model was then fitted to explore the relationship between these variables and the average rating of the movies.

```
# Fit models
model_sum_lm <- lm(avg_rating ~ genre_count + sum_good_actor_score, data = train_data)
model_avg_lm <- lm(avg_rating ~ genre_count + avg_good_actor_score, data = train_data)
print(summary(model_sum_lm))
```

```
##
## Call:
## lm(formula = avg_rating ~ genre_count + sum_good_actor_score,
##     data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62121 -0.41762  0.08475  0.41572  1.93507
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.140e+00  2.366e-02  132.68   <2e-16 ***
## genre_count           -1.847e-02  9.236e-03   -2.00   0.0456 *
```

```
## sum_good_actor_score  4.395e-04  1.572e-05   27.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7653 on 5463 degrees of freedom
## Multiple R-squared:  0.1252, Adjusted R-squared:  0.1249
## F-statistic: 390.9 on 2 and 5463 DF,  p-value: < 2.2e-16
```

```
print(summary(model_avg_lm))
```

```
##
## Call:
## lm(formula = avg_rating ~ genre_count + avg_good_actor_score,
##     data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5068 -0.4257  0.0915  0.4270  1.9301
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.144048   0.023999 131.007   <2e-16 ***
## genre_count          -0.017761   0.009371  -1.895   0.0581 .
## avg_good_actor_score  0.003664   0.000148  24.757   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7758 on 5463 degrees of freedom
## Multiple R-squared:  0.101,  Adjusted R-squared:  0.1006
## F-statistic: 306.7 on 2 and 5463 DF,  p-value: < 2.2e-16
```

```
# Predictions
test_data_sum_lm <- test_data
test_data_avg_lm <- test_data
test_data_sum_lm$predicted_rating <- predict(model_sum_lm, newdata = test_data_sum_lm)
test_data_avg_lm$predicted_rating <- predict(model_avg_lm, newdata = test_data_avg_lm)

# Evaluate the model using RMSE
mse_sum <- mean((test_data_sum_lm$predicted_rating - test_data_sum_lm$avg_rating)^2)
rmse_sum <- sqrt(mse_sum)
print(paste("Root Mean Squared Error (Sum of good actor score): ", rmse_sum))
```

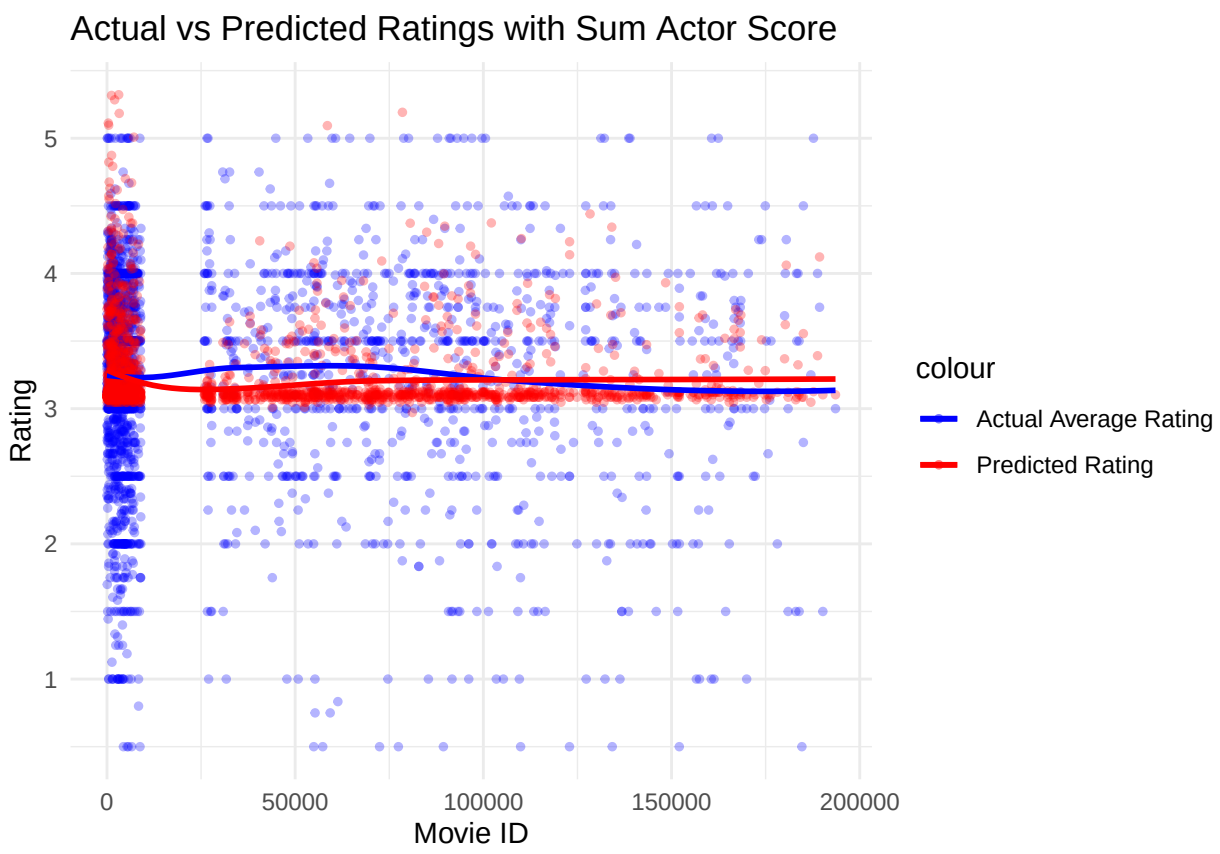
```
## [1] "Root Mean Squared Error (Sum of good actor score):  0.770956861049523"
```

```
mse_avg <- mean((test_data_avg_lm$predicted_rating - test_data_avg_lm$avg_rating)^2)
rmse_avg <- sqrt(mse_avg)
print(paste("Root Mean Squared Error (Avg of good actor score): ", rmse_avg))
```

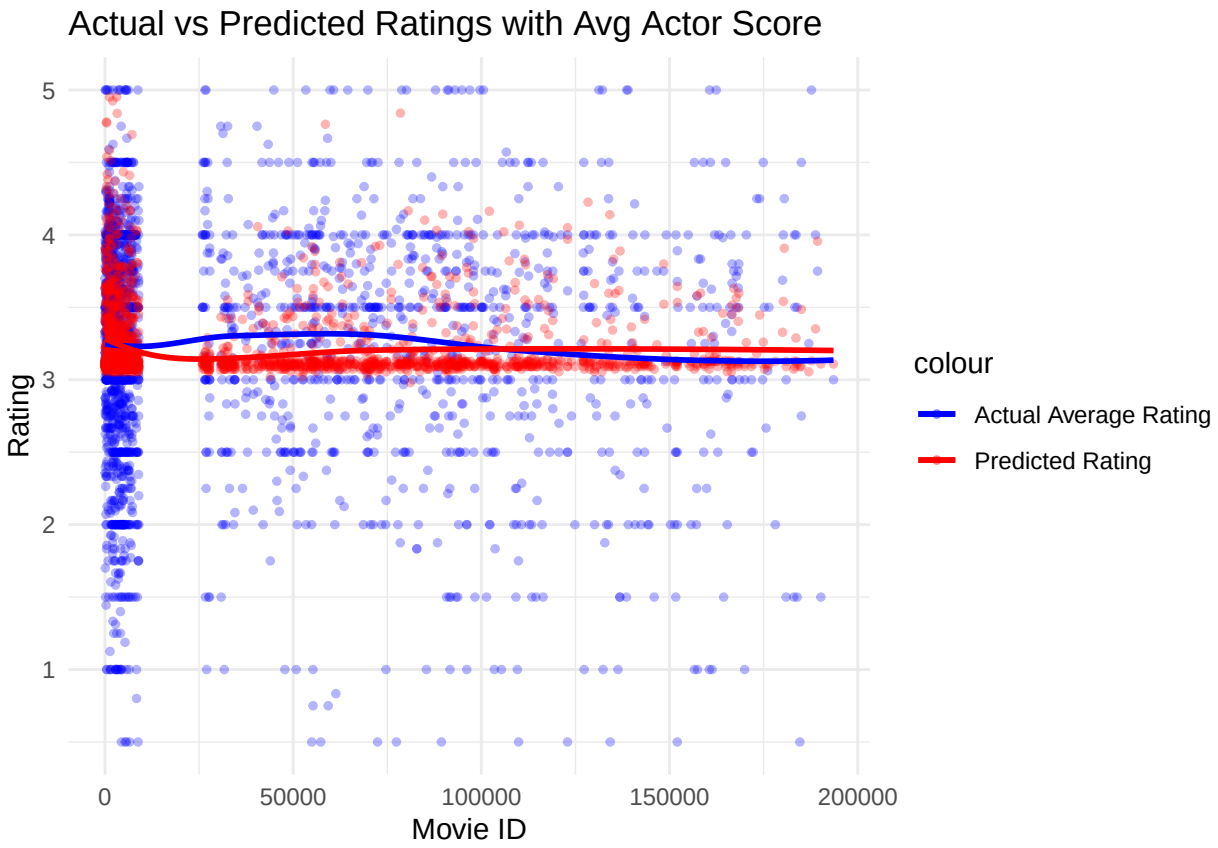
```
## [1] "Root Mean Squared Error (Avg of good actor score):  0.780882690629557"
```

```
# Plotting model_sum
ggplot(test_data_sum_lm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Sum Actor Score") +
```

```
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```

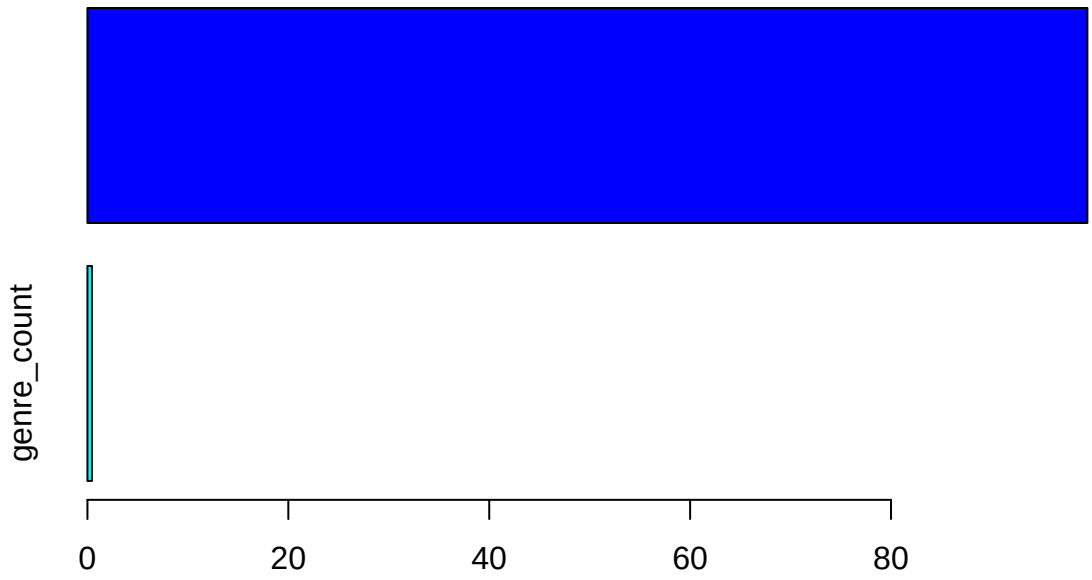



```
# Plotting model_avg
ggplot(test_data_avg_lm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Avg Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



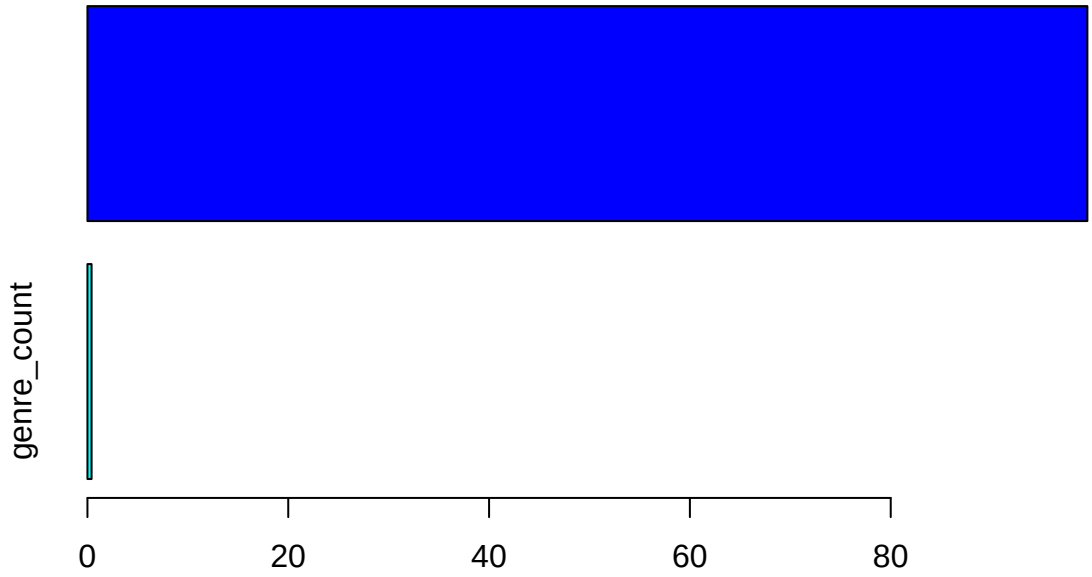
Model Fitting (Gradient Boosting Machine) - Genre Count + Good Actor Occurrences

```
# Fit models
model_sum_gbm <- gbm(avg_rating ~ genre_count + sum_good_actor_score, data = train_data, distribution =
model_avg_gbm <- gbm(avg_rating ~ genre_count + avg_good_actor_score, data = train_data, distribution =
print(summary(model_sum_gbm))
```




```
##                                     var    rel.inf
## sum_good_actor_score sum_good_actor_score 99.5503483
## genre_count                   genre_count  0.4496517
```

```
print(summary(model_avg_gbm))
```




```
##                                     var    rel.inf
## avg_good_actor_score avg_good_actor_score 99.5871317
## genre_count                   genre_count  0.4128683
```

```
# Predictions
test_data_sum_gbm <- test_data
test_data_avg_gbm <- test_data
```

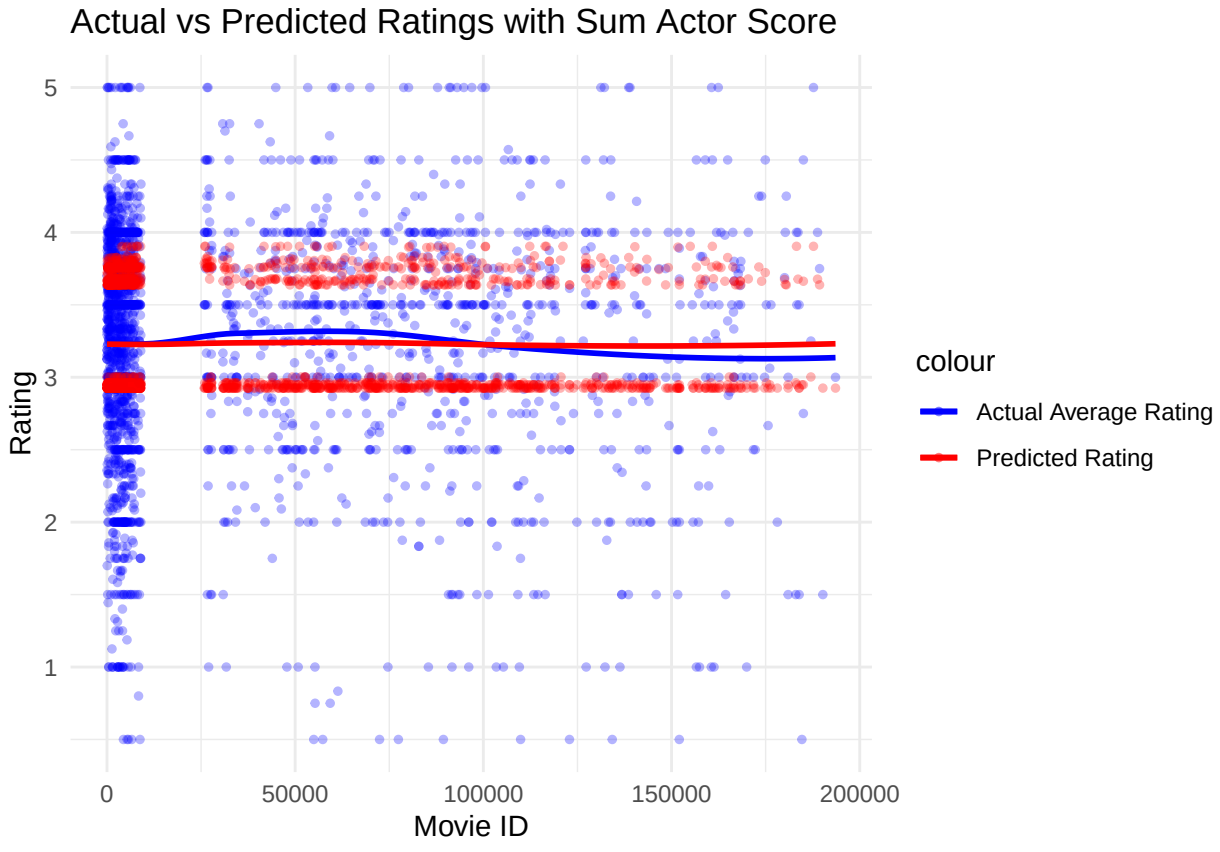
```r
test_data_sum_gbm$predicted_rating <- predict(model_sum_gbm, newdata = test_data_sum_gbm)
test_data_avg_gbm$predicted_rating <- predict(model_avg_gbm, newdata = test_data_avg_gbm)

# Evaluate the model using RMSE
mse_sum <- mean((test_data_sum_gbm$predicted_rating - test_data_sum_gbm$avg_rating)^2)
rmse_sum <- sqrt(mse_sum)
print(paste("Root Mean Squared Error: ", rmse_sum))
```

```
## [1] "Root Mean Squared Error:  0.609208372799083"
```

```r
mse_avg <- mean((test_data_avg_gbm$predicted_rating - test_data_avg_gbm$avg_rating)^2)
rmse_avg <- sqrt(mse_avg)
print(paste("Root Mean Squared Error: ", rmse_avg))
```

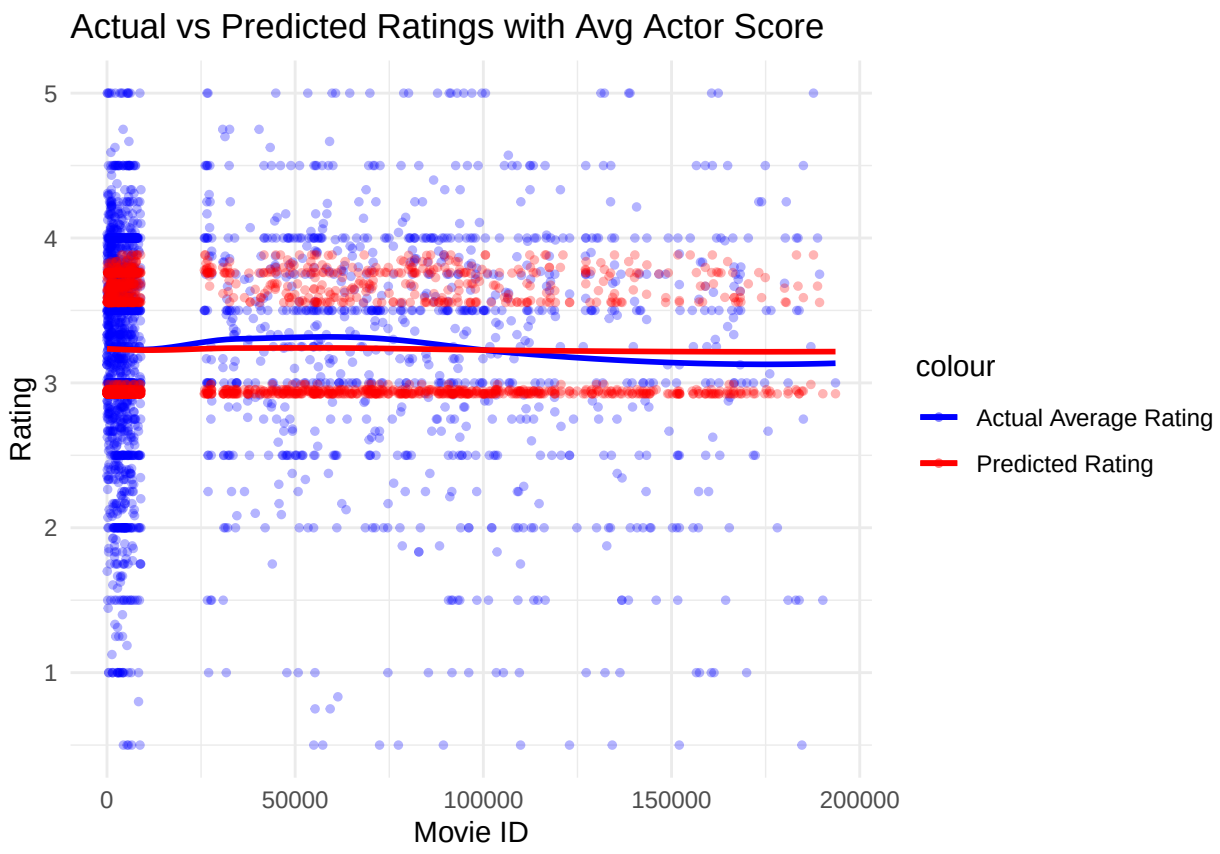
```
## [1] "Root Mean Squared Error:  0.619875896904499"
```

```r
# Plotting model_sum
ggplot(test_data_sum_gbm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Sum Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```




```r
# Plotting model_avg
ggplot(test_data_avg_gbm, aes(x = movieId)) +
```

```
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Avg Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



Model Fitting (Linear Regression) - Genre Count + Good Actor Occurrences + Decade

```
# Fit models
model_sum_decade_lm <- lm(avg_rating ~ genre_count + sum_good_actor_score + decade, data = train_data)
model_avg_decade_lm <- lm(avg_rating ~ genre_count + avg_good_actor_score + decade, data = train_data)
print(summary(model_sum_decade_lm))
```

```
##
## Call:
## lm(formula = avg_rating ~ genre_count + sum_good_actor_score +
##     decade, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.92396 -0.41470  0.07471  0.44593  2.00847
##
## Coefficients:
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.276e+01  1.355e+00   9.424  < 2e-16 ***
## genre_count          -1.584e-02  9.206e-03  -1.720   0.0855 .
## sum_good_actor_score  4.360e-04  1.566e-05  27.836  < 2e-16 ***
## decade               -4.832e-03  6.799e-04  -7.107 1.34e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7619 on 5458 degrees of freedom
##   (4 observations deleted due to missingness)
## Multiple R-squared:  0.1332, Adjusted R-squared:  0.1328
## F-statistic: 279.7 on 3 and 5458 DF,  p-value: < 2.2e-16
```

```
print(summary(model_avg_decade_lm))
```

```
##
## Call:
## lm(formula = avg_rating ~ genre_count + avg_good_actor_score +
##     decade, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4073 -0.4266  0.0814  0.4575  2.0099
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           13.5035949  1.3717793   9.844  < 2e-16 ***
## genre_count           -0.0150355  0.0093345  -1.611    0.107
## avg_good_actor_score   0.0036494  0.0001473  24.777  < 2e-16 ***
## decade                -0.0052005  0.0006886  -7.553 4.97e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.772 on 5458 degrees of freedom
##   (4 observations deleted due to missingness)
## Multiple R-squared:  0.1103, Adjusted R-squared:  0.1098
## F-statistic: 225.5 on 3 and 5458 DF,  p-value: < 2.2e-16
```

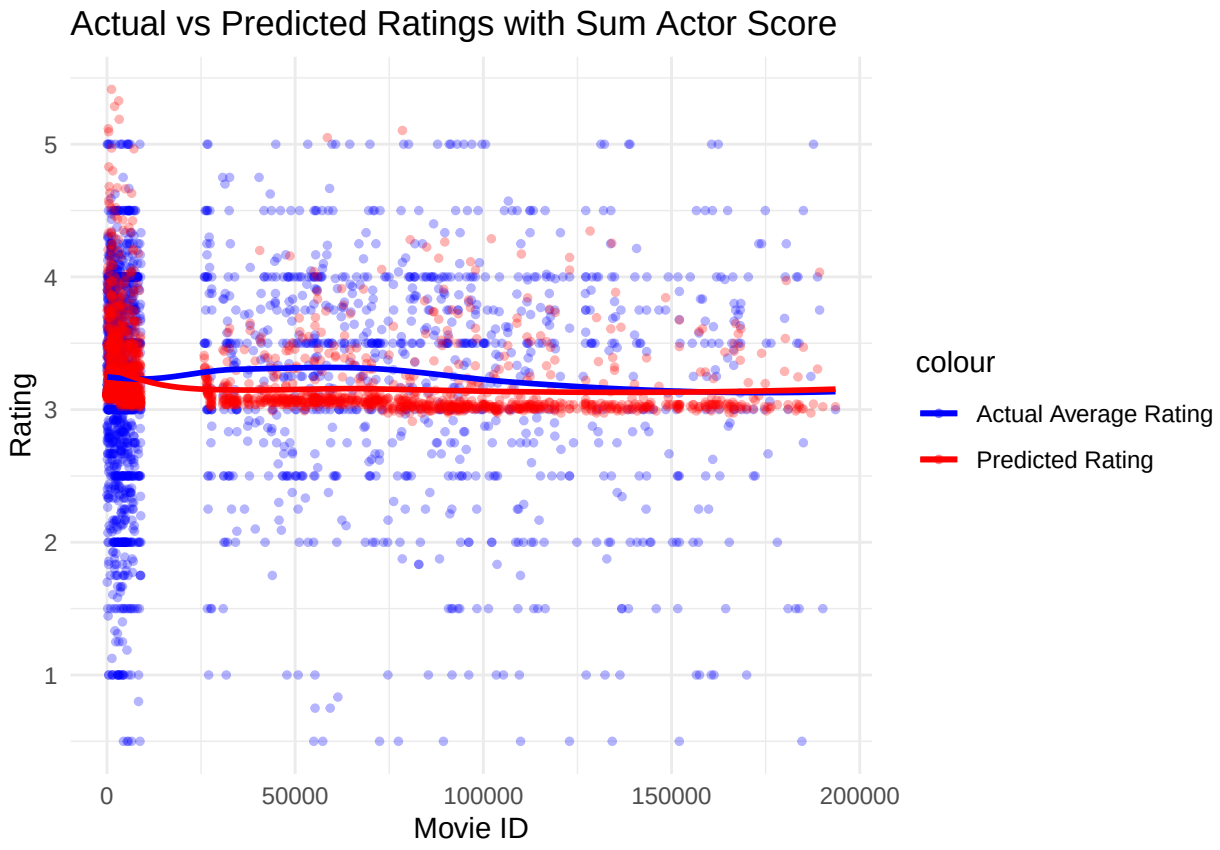
```
# Predictions
test_data_sum_decade_lm <- test_data
test_data_avg_decade_lm <- test_data
test_data_sum_decade_lm$predicted_rating <- predict(model_sum_decade_lm, newdata = test_data_sum_decade
test_data_avg_decade_lm$predicted_rating <- predict(model_avg_decade_lm, newdata = test_data_avg_decade

# Evaluate the model using RMSE
mse_sum <- mean((test_data_sum_decade_lm$predicted_rating - test_data_sum_decade_lm$avg_rating)^2, na.r
rmse_sum <- sqrt(mse_sum)
print(paste("Root Mean Squared Error (Sum of good actor score): ", rmse_sum))
```

```
## [1] "Root Mean Squared Error (Sum of good actor score):  0.763751310337971"
```

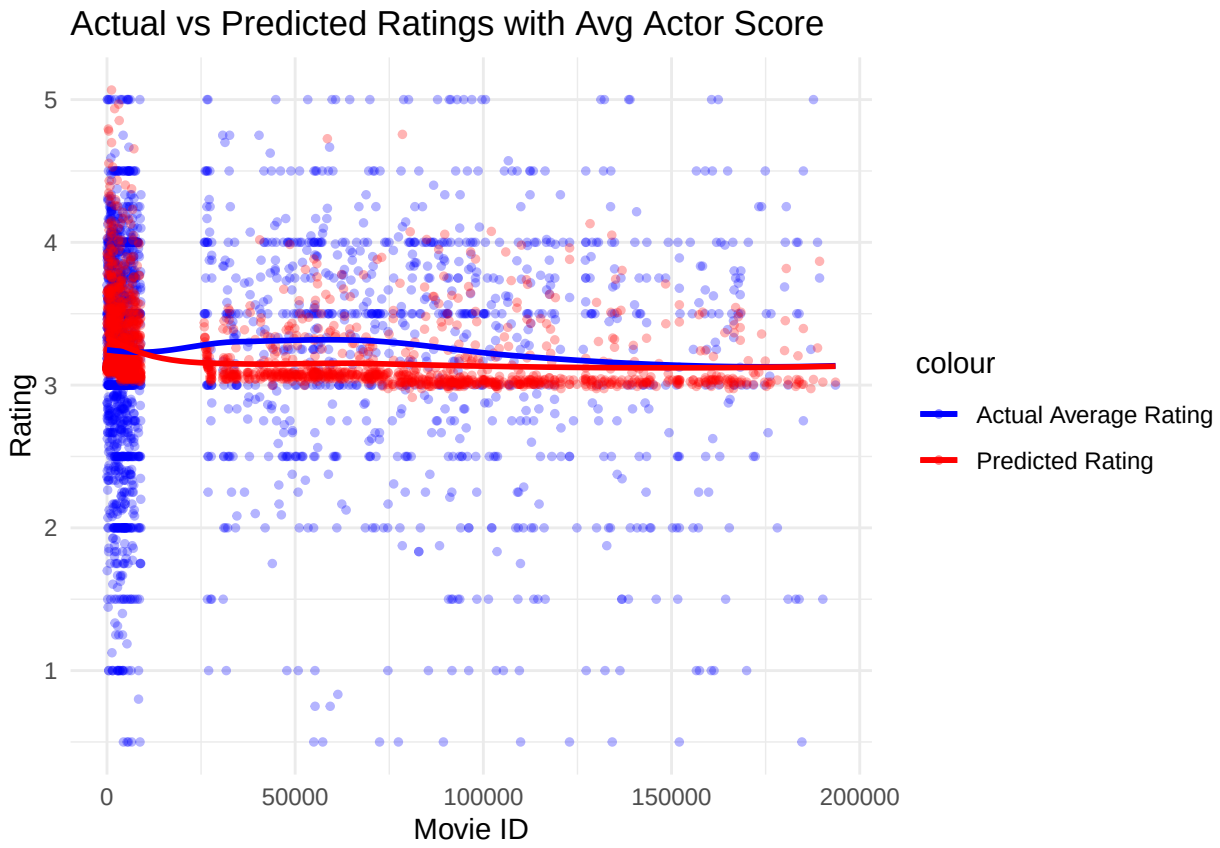
```
mse_avg <- mean((test_data_avg_decade_lm$predicted_rating - test_data_avg_decade_lm$avg_rating)^2, na.r
rmse_avg <- sqrt(mse_avg)
print(paste("Root Mean Squared Error (Avg of good actor score): ", rmse_avg))
```

```
## [1] "Root Mean Squared Error (Avg of good actor score):  0.7731642363765"
```

```r
# Plotting model_sum
ggplot(test_data_sum_decade_lm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Sum Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```

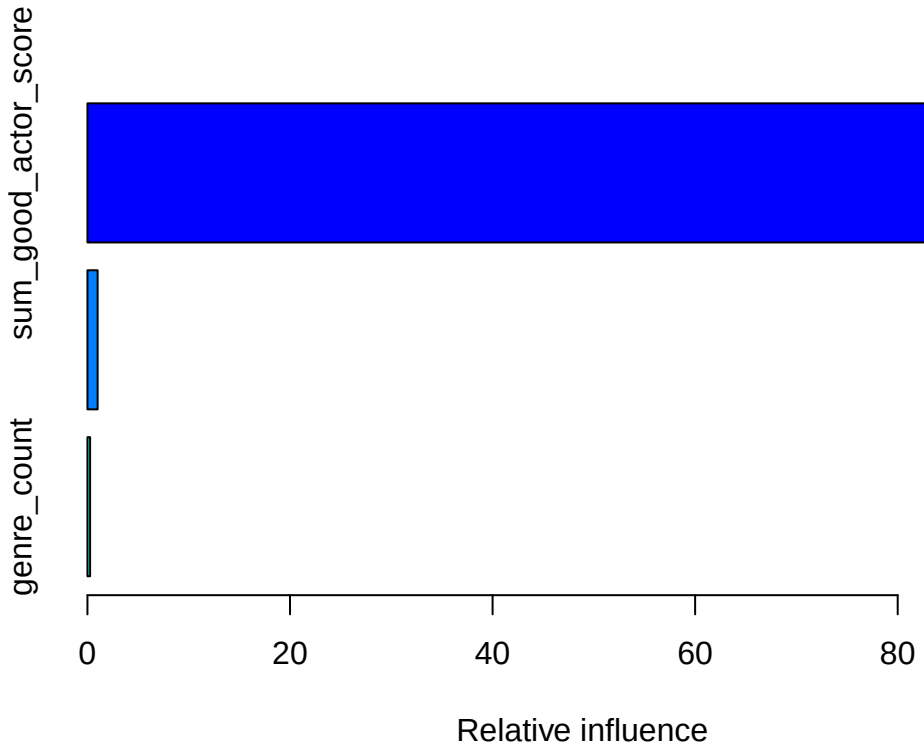



```r
# Plotting model_avg
ggplot(test_data_avg_decade_lm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Avg Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



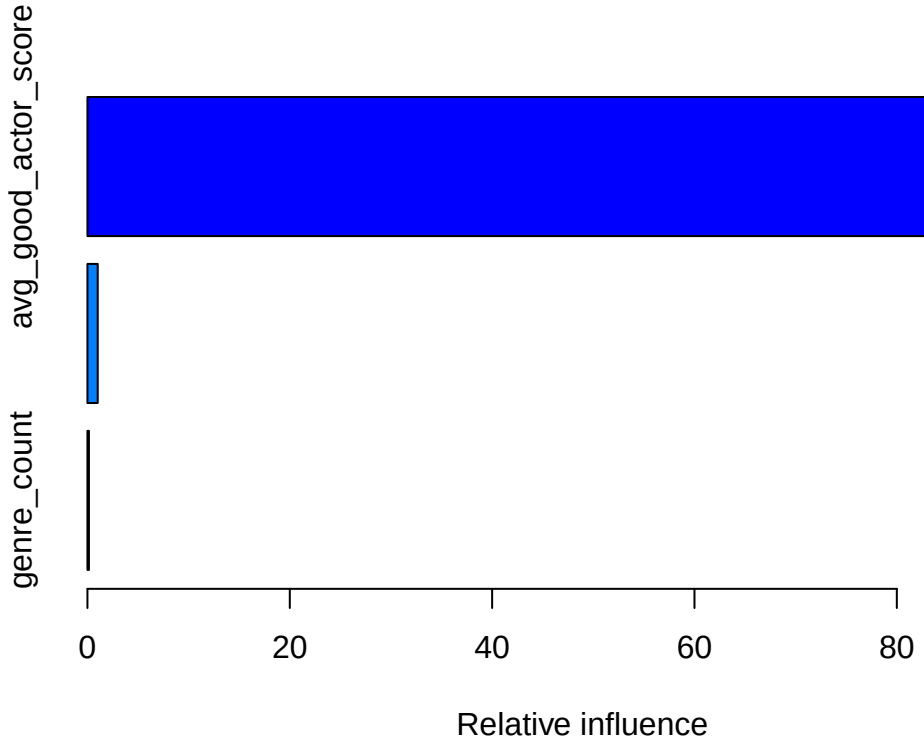
Model Fitting (Gradient Boosting Machine) - Genre Count + Good Actor Occurrences + Decade

```
# Fit models
model_sum_decade_gbm <- gbm(avg_rating ~ genre_count + sum_good_actor_score + decade, data = train_data
model_avg_decade_gbm <- gbm(avg_rating ~ genre_count + avg_good_actor_score + decade, data = train_data
print(summary(model_sum_decade_gbm))
```




```
##                                     var    rel.inf
## sum_good_actor_score sum_good_actor_score 98.7312361
## decade                           decade  1.0058186
## genre_count                   genre_count  0.2629452
```

```
print(summary(model_avg_decade_gbm))
```



```
##                                     var    rel.inf
## avg_good_actor_score avg_good_actor_score 98.8345347
## decade                           decade  1.0184652
## genre_count                 genre_count  0.1470001
```

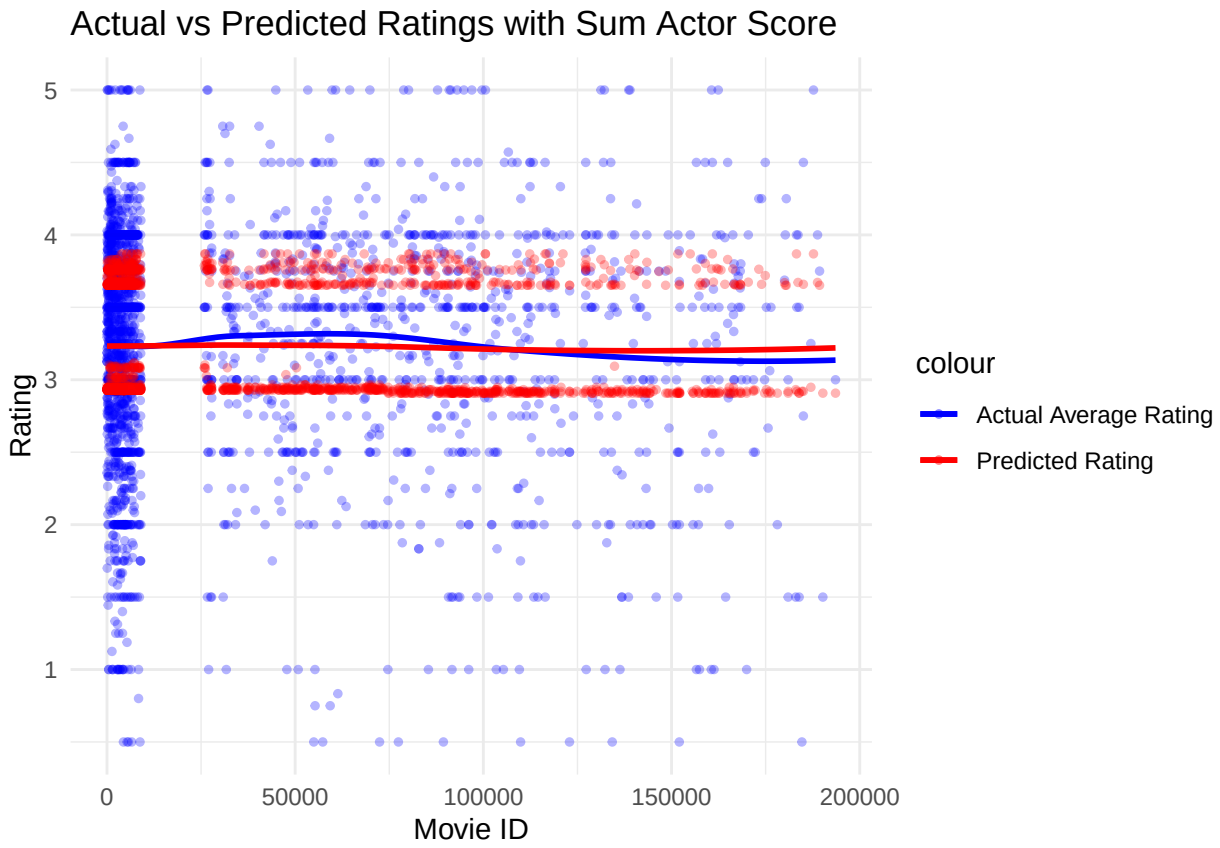
```r
# Predictions
test_data_sum_decade_gbm <- test_data
test_data_avg_decade_gbm <- test_data
test_data_sum_decade_gbm$predicted_rating <- predict(model_sum_decade_gbm, newdata = test_data_sum_decad
test_data_avg_decade_gbm$predicted_rating <- predict(model_avg_decade_gbm, newdata = test_data_avg_decad

# Evaluate the model using RMSE
mse_sum <- mean((test_data_sum_decade_gbm$predicted_rating - test_data_sum_decade_gbm$avg_rating)^2, na
rmse_sum <- sqrt(mse_sum)
print(paste("Root Mean Squared Error (Sum of good actor score): ", rmse_sum))
```

```
## [1] "Root Mean Squared Error (Sum of good actor score):  0.606729927216438"
```

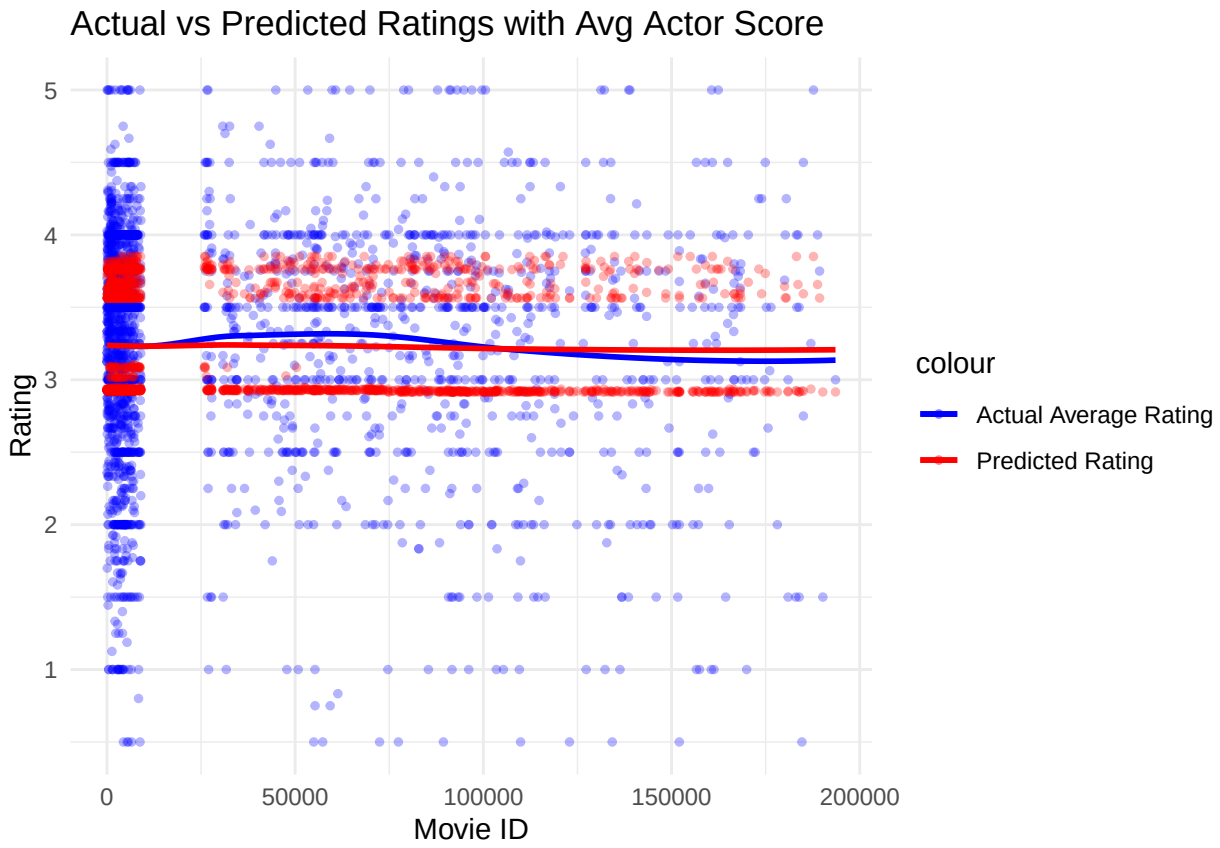
```r
mse_avg <- mean((test_data_avg_decade_gbm$predicted_rating - test_data_avg_decade_gbm$avg_rating)^2, na
rmse_avg <- sqrt(mse_avg)
print(paste("Root Mean Squared Error (Avg of good actor score): ", rmse_avg))
```

```
## [1] "Root Mean Squared Error (Avg of good actor score):  0.618160514508124"
```

```r
# Plotting model_sum
ggplot(test_data_sum_decade_gbm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Sum Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```




```
# Plotting model_avg
ggplot(test_data_avg_decade_gbm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Avg Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



Model Fitting (Linear Regression) - Genre Count + Good Actor Occurrences + WorldWide Box Office

```
# Fit models
model_sum_box_lm <- lm(avg_rating ~ genre_count + sum_good_actor_score + WorldWide, data = train_data)
model_avg_box_lm <- lm(avg_rating ~ genre_count + avg_good_actor_score + WorldWide, data = train_data)
print(summary(model_sum_box_lm))
```

```
##
## Call:
## lm(formula = avg_rating ~ genre_count + sum_good_actor_score +
##     WorldWide, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62568 -0.40546  0.09726  0.42160  1.88883
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.128e+00  2.365e-02 132.275  < 2e-16 ***
## genre_count           -2.297e-03  9.541e-03  -0.241     0.81
## sum_good_actor_score   4.674e-04  1.626e-05  28.748  < 2e-16 ***
## WorldWide             -4.862e-10  7.577e-11  -6.416 1.51e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.7625 on 5462 degrees of freedom
## Multiple R-squared:  0.1317, Adjusted R-squared:  0.1313
## F-statistic: 276.3 on 3 and 5462 DF,  p-value: < 2.2e-16
```

```
print(summary(model_avg_box_lm))
```

```
##
## Call:
## lm(formula = avg_rating ~ genre_count + avg_good_actor_score +
##     WorldWide, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7416 -0.4167  0.1068  0.4332  1.8897
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.132e+00  2.399e-02 130.514  < 2e-16 ***
## genre_count          -1.715e-03  9.673e-03  -0.177    0.859
## avg_good_actor_score  3.955e-03  1.544e-04  25.612  < 2e-16 ***
## WorldWide            -4.924e-10  7.751e-11  -6.353 2.28e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.773 on 5462 degrees of freedom
## Multiple R-squared:  0.1076, Adjusted R-squared:  0.1071
## F-statistic: 219.4 on 3 and 5462 DF,  p-value: < 2.2e-16
```

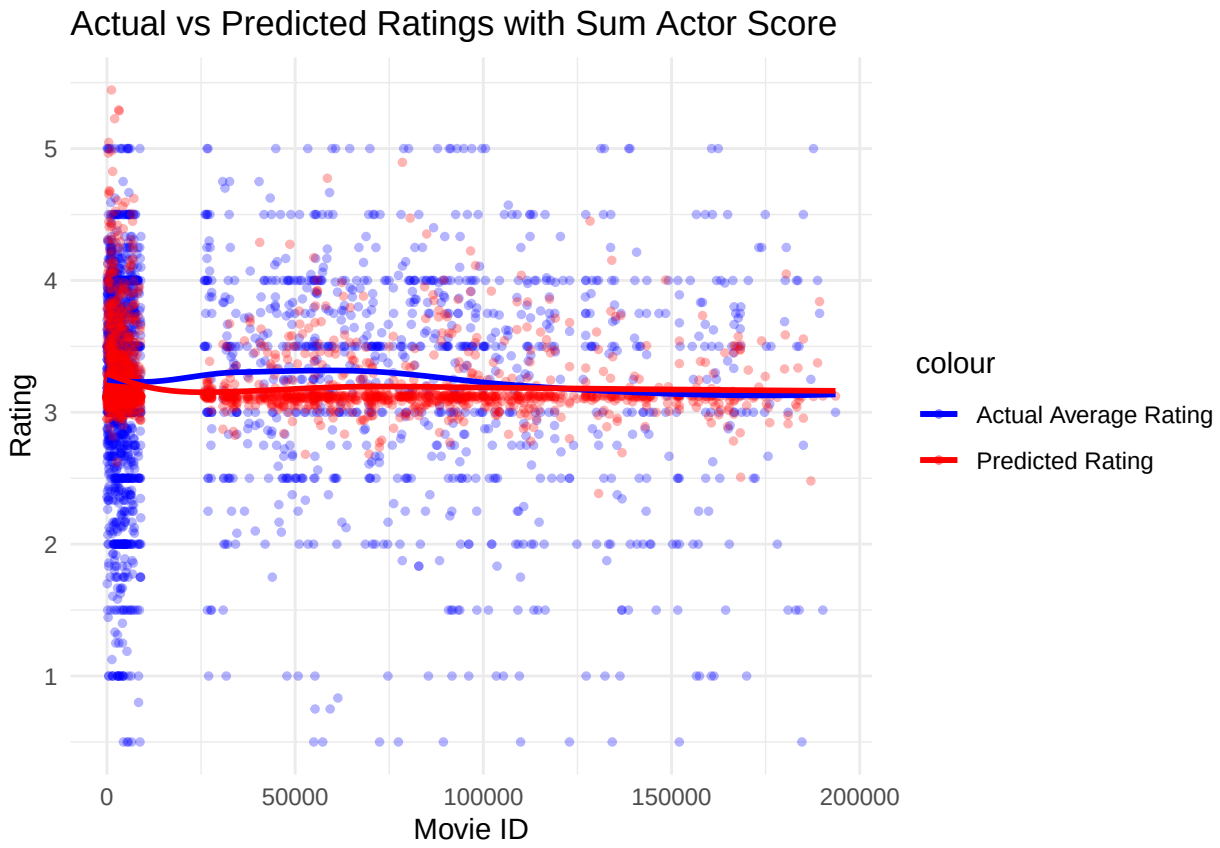
```
# Predictions
test_data_sum_box_lm <- test_data
test_data_avg_box_lm <- test_data
test_data_sum_box_lm$predicted_rating <- predict(model_sum_box_lm, newdata = test_data_sum_box_lm)
test_data_avg_box_lm$predicted_rating <- predict(model_avg_box_lm, newdata = test_data_avg_box_lm)

# Evaluate the model using RMSE
mse_sum <- mean((test_data_sum_box_lm$predicted_rating - test_data_sum_box_lm$avg_rating)^2)
rmse_sum <- sqrt(mse_sum)
print(paste("Root Mean Squared Error (Sum of good actor score): ", rmse_sum))
```

```
## [1] "Root Mean Squared Error (Sum of good actor score):  0.765875784096547"
```

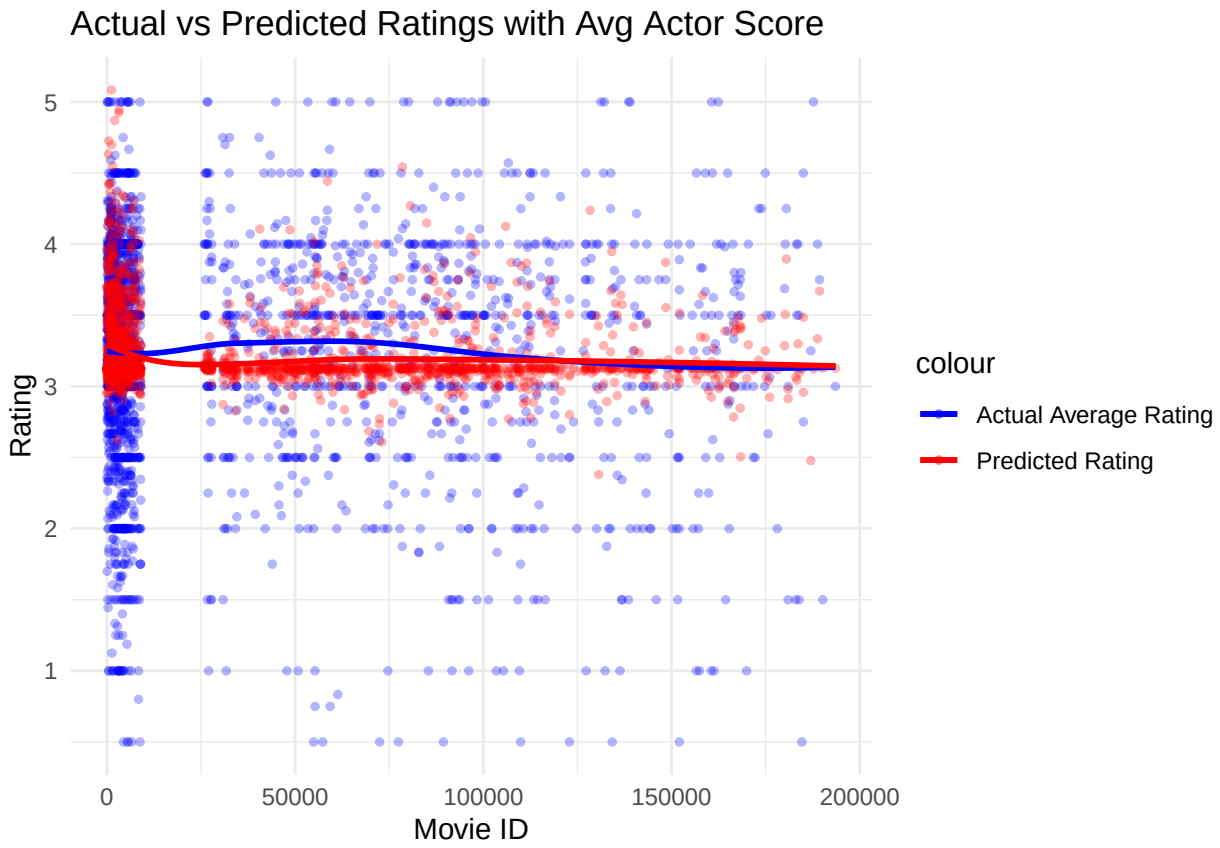
```
mse_avg <- mean((test_data_avg_box_lm$predicted_rating - test_data_avg_box_lm$avg_rating)^2)
rmse_avg <- sqrt(mse_avg)
print(paste("Root Mean Squared Error (Avg of good actor score): ", rmse_avg))
```

```
## [1] "Root Mean Squared Error (Avg of good actor score):  0.7766738349292"
```

```
# Plotting model_sum
ggplot(test_data_sum_box_lm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Sum Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```

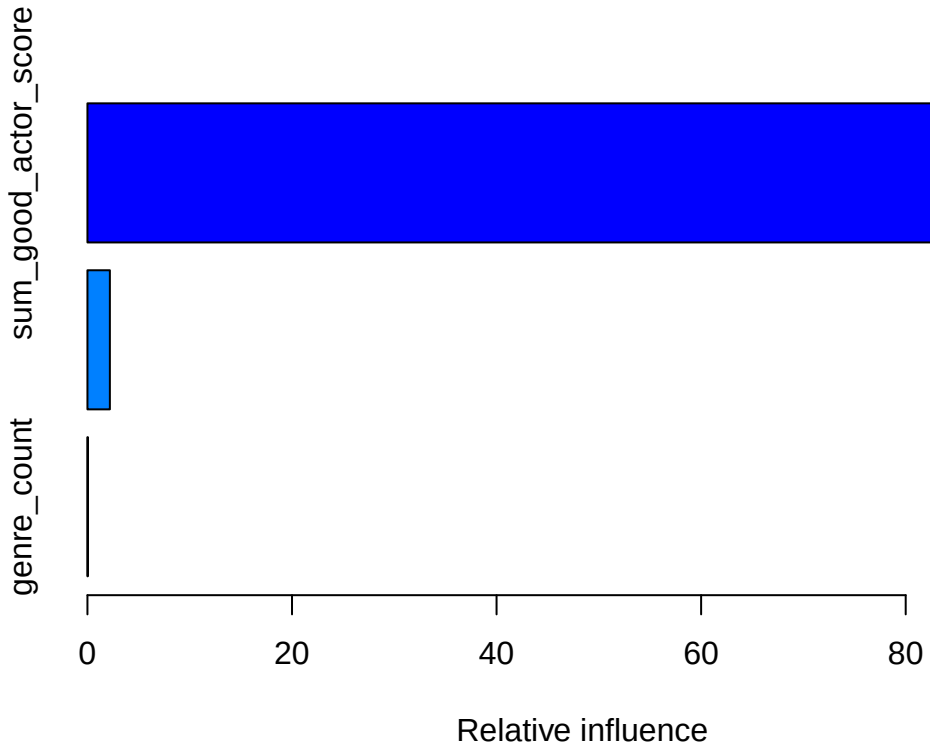



```
# Plotting model_avg
ggplot(test_data_avg_box_lm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Avg Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



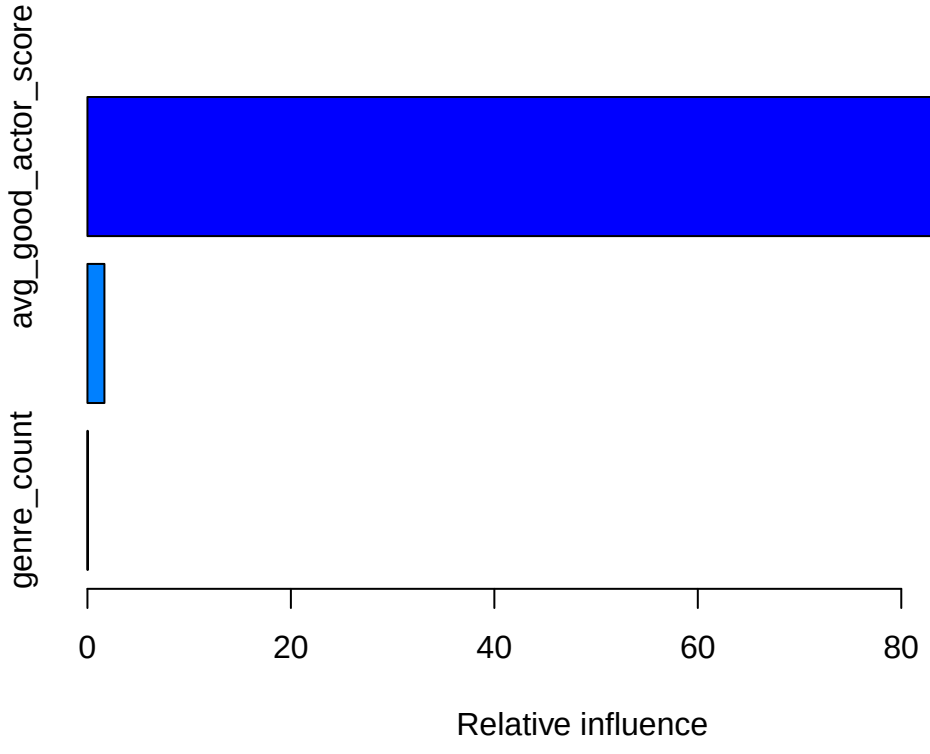
Model Fitting (Gradient Boosting Machine) - Genre Count + Good Actor Occurrences + WorldWide Box Office

```
# Fit models
model_sum_box_gbm <- gbm(avg_rating ~ genre_count + sum_good_actor_score + WorldWide, data = train_data
model_avg_box_gbm <- gbm(avg_rating ~ genre_count + avg_good_actor_score + WorldWide, data = train_data
print(summary(model_sum_box_gbm))
```




```
##                                     var     rel.inf
## sum_good_actor_score sum_good_actor_score 97.76738408
## WorldWide                     WorldWide  2.19202817
## genre_count                 genre_count  0.04058776
```

```
print(summary(model_avg_box_gbm))
```



```
##                                     var     rel.inf
## avg_good_actor_score avg_good_actor_score 98.29499195
## WorldWide                     WorldWide  1.66957749
## genre_count                 genre_count  0.03543056
```

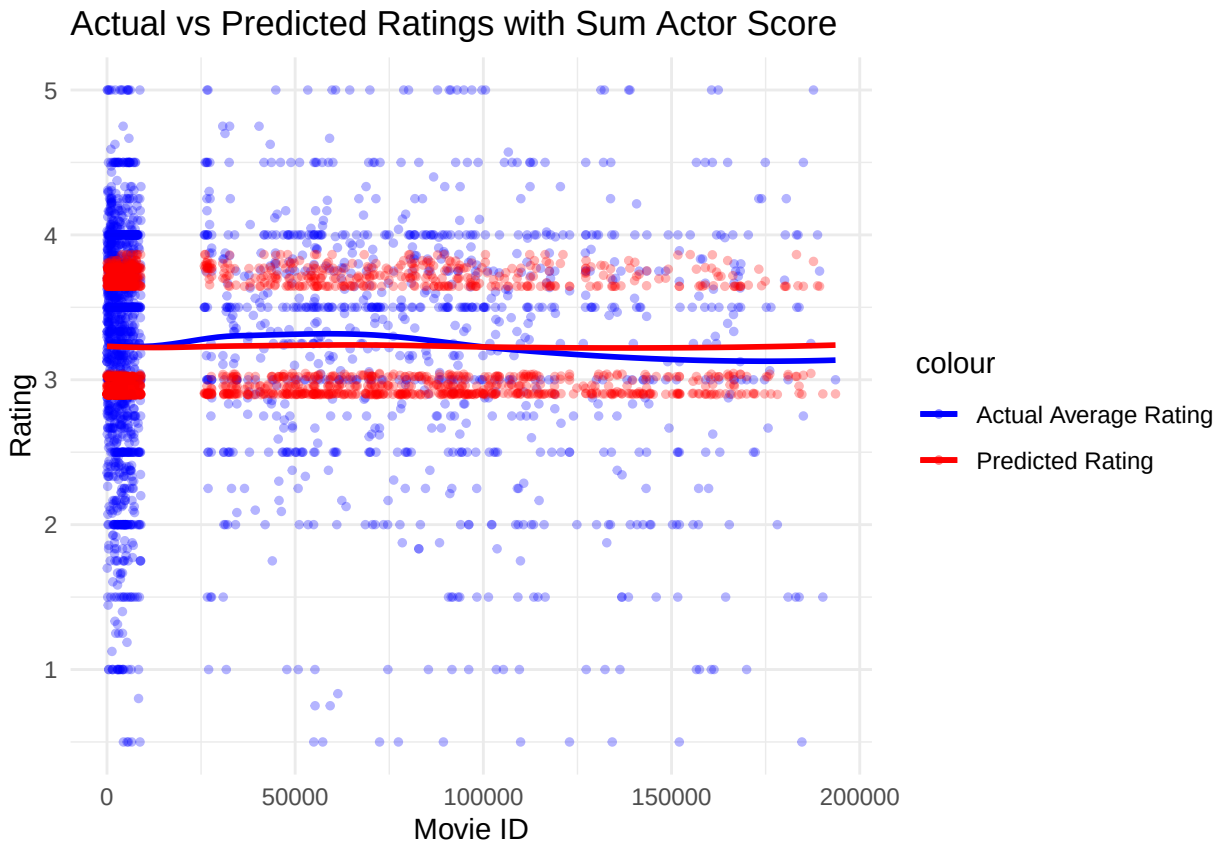
```r
# Predictions
test_data_sum_box_gbm <- test_data
test_data_avg_box_gbm <- test_data
test_data_sum_box_gbm$predicted_rating <- predict(model_sum_box_gbm, newdata = test_data_sum_box_gbm)
test_data_avg_box_gbm$predicted_rating <- predict(model_avg_box_gbm, newdata = test_data_avg_box_gbm)

# Evaluate the model using RMSE
mse_sum <- mean((test_data_sum_box_gbm$predicted_rating - test_data_sum_box_gbm$avg_rating)^2)
rmse_sum <- sqrt(mse_sum)
print(paste("Root Mean Squared Error: ", rmse_sum))
```

```
## [1] "Root Mean Squared Error:  0.607852930486117"
```

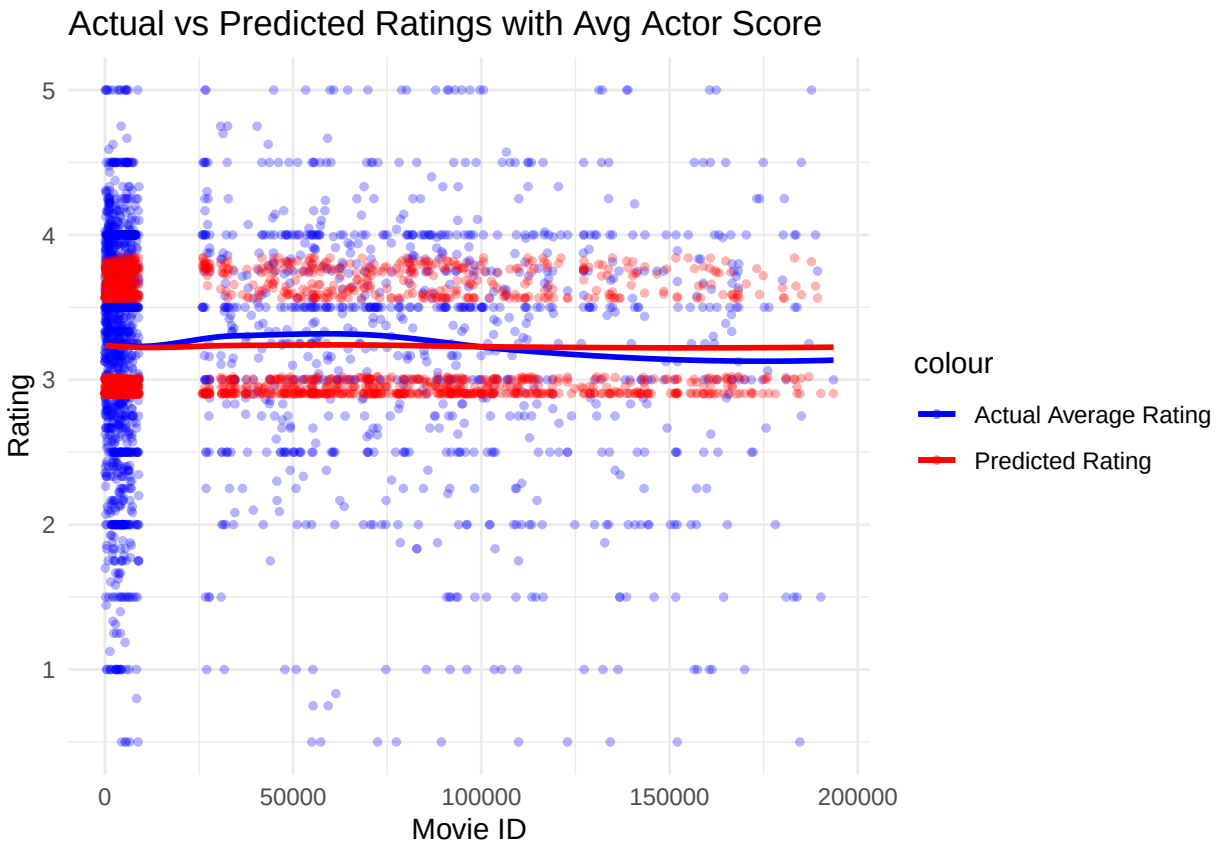
```r
mse_avg <- mean((test_data_avg_box_gbm$predicted_rating - test_data_avg_box_gbm$avg_rating)^2)
rmse_avg <- sqrt(mse_avg)
print(paste("Root Mean Squared Error: ", rmse_avg))
```

```
## [1] "Root Mean Squared Error:  0.619013051033723"
```

```r
# Plotting model_sum
ggplot(test_data_sum_box_gbm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Sum Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



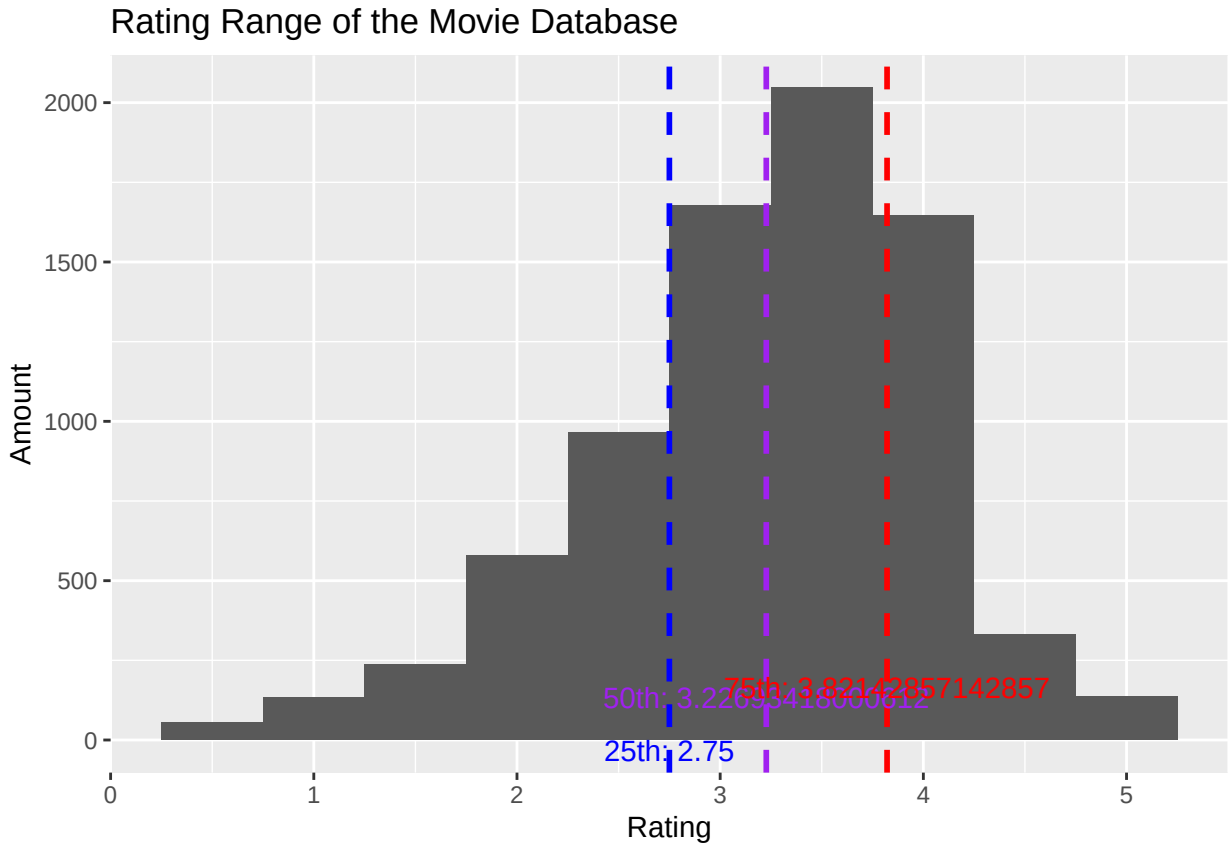

```r
# Plotting model_avg
ggplot(test_data_avg_box_gbm, aes(x = movieId)) +
  geom_point(aes(y = avg_rating, color = "Actual Average Rating"), size = 1, alpha = 0.3) +
  geom_point(aes(y = predicted_rating, color = "Predicted Rating"), size = 1, alpha = 0.3) +
  geom_smooth(aes(y = avg_rating, color = "Actual Average Rating"), method = "loess", se = FALSE) +
  geom_smooth(aes(y = predicted_rating, color = "Predicted Rating"), method = "loess", se = FALSE) +
  labs(x = "Movie ID", y = "Rating", title = "Actual vs Predicted Ratings with Avg Actor Score") +
  scale_color_manual(values = c("Actual Average Rating" = "blue", "Predicted Rating" = "red")) +
  theme_minimal()
```



Graph Plotting

To explore our research questions, we have created several plots below.

```
# Analyzing the rating range and distributions of the movies
q25 <- quantile(merged_data$avg_rating, probs = 0.25)
q50 <- mean(merged_data$avg_rating)
q75 <- quantile(merged_data$avg_rating, probs = 0.75)

ggplot(merged_data, aes(x = avg_rating)) +
  geom_histogram(binwidth=0.5) +
  geom_vline(xintercept = q25, linetype = "dashed", color = "blue", size=1) +
  geom_vline(xintercept = q75, linetype = "dashed", color = "red", size=1) +
  geom_vline(xintercept = q50, linetype = "dashed", color = 'purple', size=1) +
  labs(title = "Rating Range of the Movie Database", x = "Rating", y = "Amount") +
  annotate("text", x = q25, y = 0, label = paste("25th:", q25), vjust = 1, color = "blue") +
  annotate("text", x = q50, y = 0, label = paste("50th:", q50), vjust = -1.5, color = "purple") +
  annotate("text", x = q75, y = 0, label = paste("75th:", q75), vjust = -2, color = "red")
```



Discussion

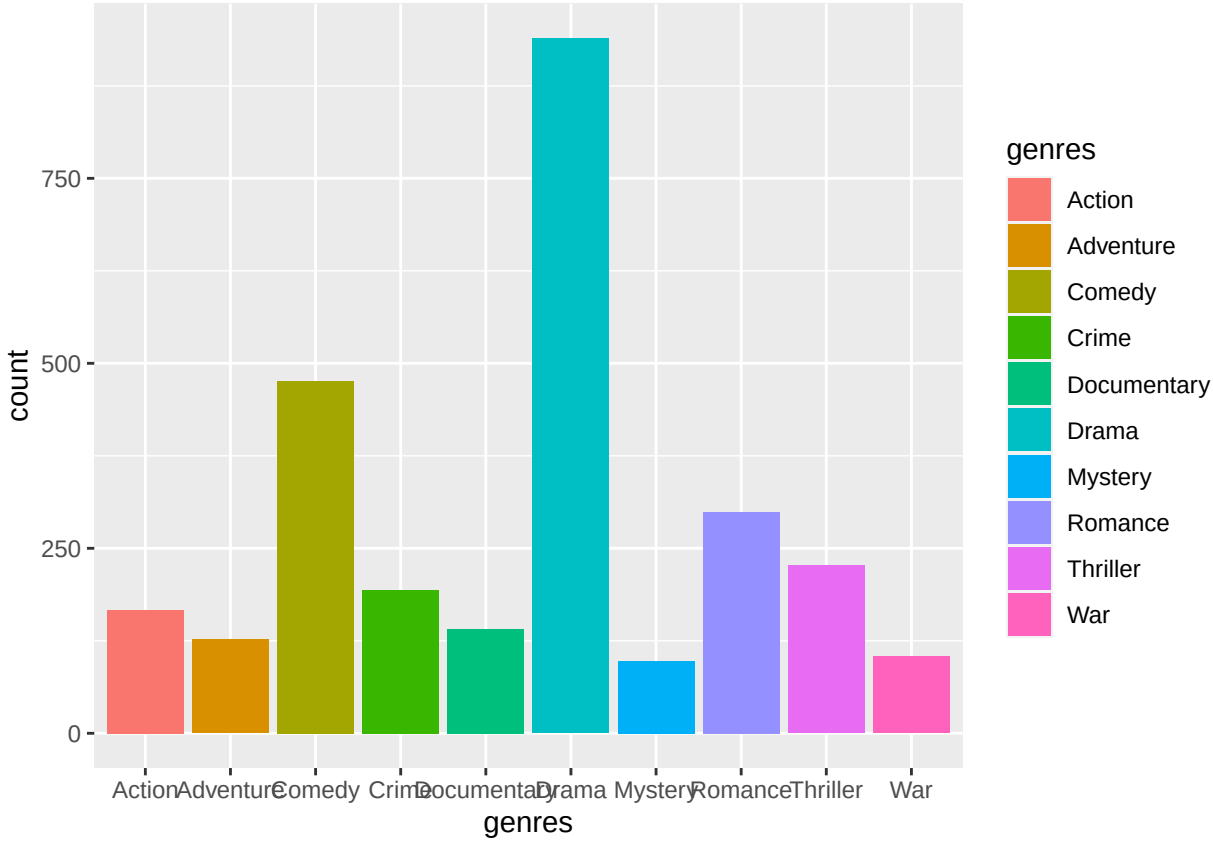
The plot provides some clear insights:

1. **Central Tendency:** The noticeable peak in the bar height around the median rating of approximately 3.5 suggests that a significant number of movies receive average scores. This concentration around the median indicates that while some films receive exceptionally high or low ratings, most are rated moderately. This pattern may reflect either a consistent quality in the movie selection or a general tendency of the audience to rate movies as average.
2. **Diversity in Ratings:** The considerable gap between the 25th percentile (blue dashed line) and the 75th percentile (red dashed line) points to a broad distribution of ratings. This is further evidenced by the bar chart, which shows most ratings falling between 3 and 4. Consequently, our analysis can be guided in this direction to ascertain whether a movie is likely to be mediocre (with a rating of around 3) or phenomenal (with a rating of more than 3).

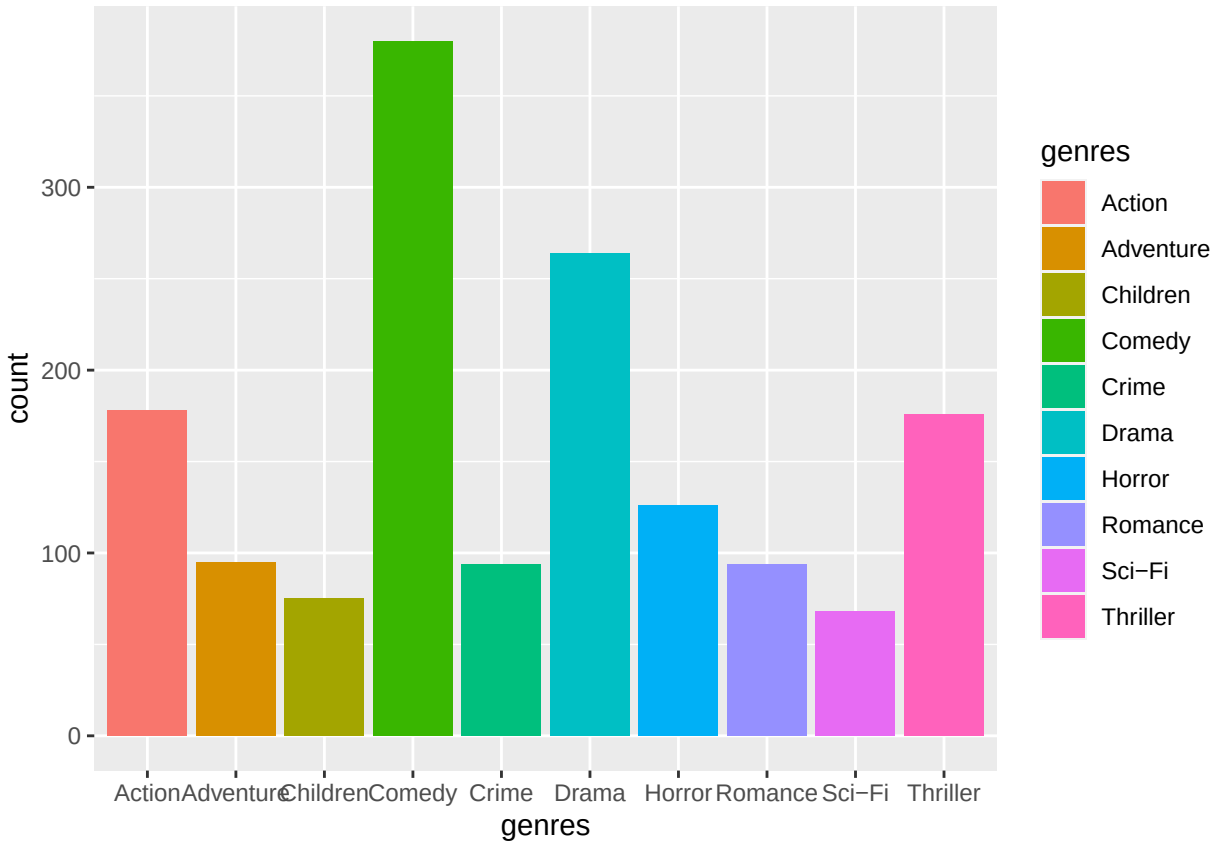
These observations lead our analysis to further investigate which features may influence the final ratings.

```
genres_ratings_10 <- merged_data %>%
  group_by(genres) %>%
  summarise(average_rating = mean(avg_rating, na.rm = TRUE)) %>%
  arrange(desc(average_rating)) %>%
  slice_head(n=10)

movies_over_4_rating <- merged_data %>%
  filter(avg_rating >= 4)
movies_under_2_rating <- merged_data %>%
  filter(avg_rating <=2)
```

```r
genres_separated <- movies_over_4_rating %>%
  separate_rows(genres, sep = "\\|")
genres_separated_low <- movies_under_2_rating %>%
  separate_rows(genres, sep="\\|")

genre_counts <- genres_separated %>%
  count(genres, name = "count") %>%
  arrange(desc(count)) %>%
  slice_head(n=10)
genre_counts_low <- genres_separated_low %>%
  count(genres, name="count") %>%
  arrange(desc(count)) %>%
  slice_head(n=10)

ggplot(genre_counts, aes(x = genres, y = count, fill=genres )) +
     geom_bar(stat="identity")
```




```r
ggplot(genre_counts_low, aes(x = genres, y = count,fill=genres )) +
     geom_bar(stat="identity")
```



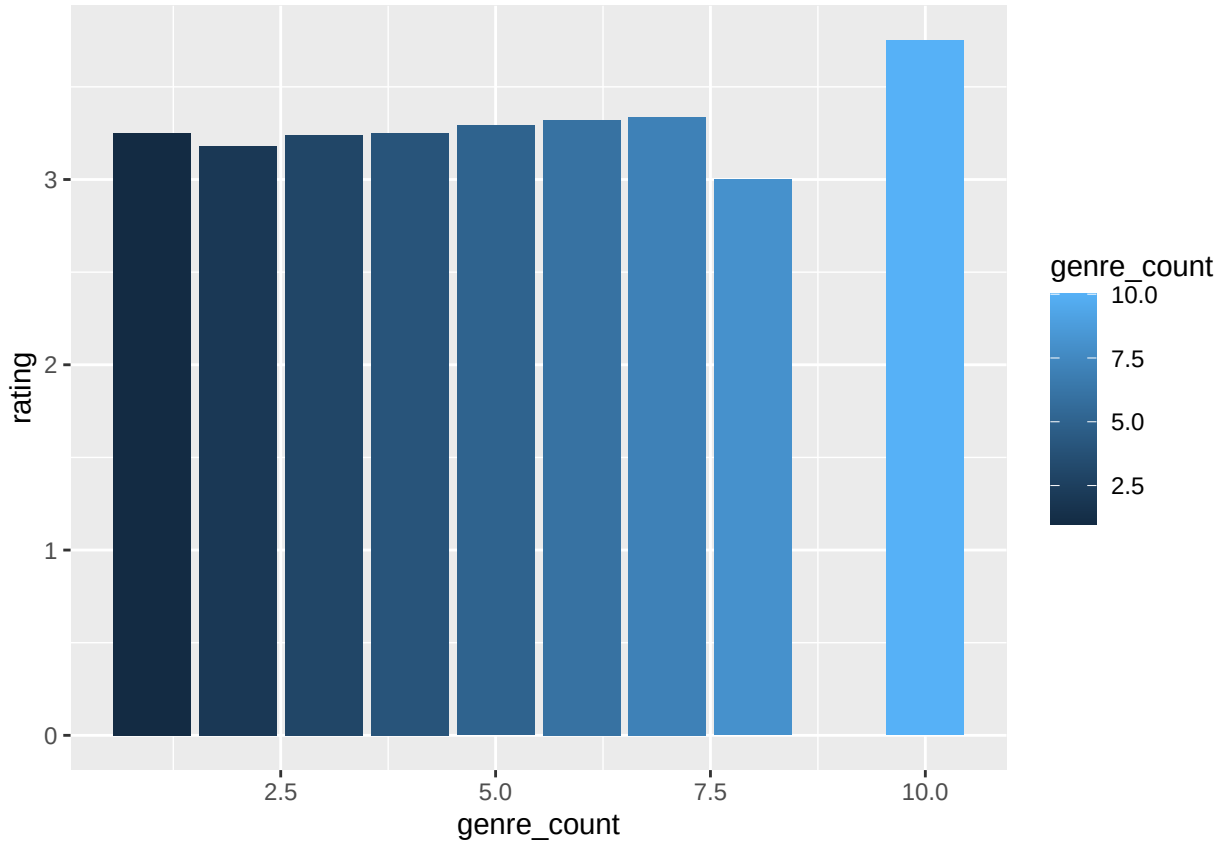
Discussion

The bar chart illustrates the distribution of movies with ratings above 4 across different genres. Drama stands out prominently, indicating a strong preference for this genre among highly-rated movies. The Documentary and Comedy genres also have significant representation, suggesting their appeal in high-quality films. In contrast, genres like Action, Animation, Crime, Horror, Romance, and Thriller have fewer movies with high ratings, with Horror being the least represented. This suggests that audiences and critics may favor the storytelling and content often associated with Drama and Documentaries over the characteristics typically found in the other mentioned genres for highly-rated films.

The bar graph showing genre counts for movies with ratings of 2 or less reveals a different trend. Drama, which was dominant in higher ratings, is less prevalent here, suggesting it is generally well-received. Comedy, despite being popular in higher ratings, is the most common genre among lower-rated movies, indicating a divided reception. Action and Thriller genres also appear more frequently in this lower rating spectrum, possibly reflecting critical disdain for certain films within these genres. Sci-Fi and Romance have moderate counts, while Children's movies are the least represented, suggesting these genres may have moderate reception. Comparing this to the summary for movies rated 4 and above, there is a noticeable contrast: genres that excel in higher ratings, such as Drama and Documentary, are less common among lower-rated movies. This could imply that well-executed dramas and documentaries resonate well with audiences, whereas comedies can be hit-or-miss. Action and Thriller genres, while not as common among top-rated movies, appear more frequently at the lower end of the ratings spectrum, which may indicate a consistent lower critical reception or a split in audience preference.

```
genrecount_rating <- merged_data %>%
  group_by(genre_count) %>%
  summarise(rating = mean(avg_rating))

ggplot(genrecount_rating, aes(x = genre_count, y=rating, fill=genre_count )) +
```

```
    geom_bar(stat="identity")
```



Discussion

The bar chart seems to illustrate the relationship between the number of genres a movie spans and its average rating. The trend indicates that movies covering a broader range of genres do not necessarily garner higher average ratings. Notably, films with a single genre focus tend to receive moderate ratings. There is a distinct peak for movies that encompass around ten genres, which achieve the highest average ratings. This suggests a potential sweet spot where a wide diversity of genres correlates with greater appreciation from viewers or critics. However, the ratings slightly decline for movies with fewer genres, around seven to nine, before peaking. This pattern could suggest that beyond a certain threshold, adding more genres does not enhance a movie's reception and may even detract from its overall quality or coherence.

Future Work

Dataset

For future analyses, we might consider incorporating additional datasets or conducting further investigations by Python crawler to address our research questions more comprehensively:

1. **Cast Data:** We have already utilized a Python crawler to obtain cast data and will continue to employ this technique to gather more comprehensive information about the actors and actresses involved in the films. This data is crucial for exploring the impact of star power on movie ratings and success.
2. **Temporal Trends:** Analyzing changes in genre popularity and movie ratings over time could provide insights into evolving audience preferences and industry trends.
3. **Revenue and Budget Data:** Integrating data on movie revenues and budgets could help us explore the financial success of different genres and the impact of star power on a movie's commercial performance.

4. **Social Media and Marketing Data:** Investigating the role of social media buzz and marketing campaigns in influencing movie popularity and ratings could offer a more holistic view of what drives audience interest and engagement.
5. **Audience Demographics:** Examining how different demographic groups (e.g., age, gender, region) respond to various genres and cast members could reveal targeted strategies for maximizing a movie's appeal.

By expanding our dataset and employing a variety of analytical approaches, including continued Python crawling for additional cast data and other relevant information, we can deepen our understanding of the factors that influence movie ratings, genre popularity, and overall success in the film industry.

Model

Currently, our primary focus has been on feature extraction and analysis. We have employed a linear regression model and a gradient boosting machine to examine the relationships between various variables. Moving forward, we intend to explore a range of modeling approaches to gain a deeper understanding of the factors influencing movie ratings and success. By constructing and comparing different models, we aim to identify more nuanced relationships and potentially uncover new insights into the dynamics of the film industry.

Conclusion

In this proposal, we have undertaken an extensive analysis of movie ratings and their influencing factors, using datasets from MovieLens and IMDb. We have explored the impact of genre diversity, cast popularity, and other factors on movie ratings. Our findings suggest that while a broader range of genres and star power can positively influence ratings, the relationship is complex and warrants further investigation.