Feature Analysis of Movie Attributes

Madhusudan Malhar Deshpande, Arif Sarfaraz Waghbakriwala, Reem Ghabayen, Aanand Dhandapani

Summary:

A Movie can be classified as a successful movie or a ' hit' based on the revenue it earns for the budget put into its making. Of course, the story of the Movie plays a vital role in determining that. However, there could be hidden patterns in the movie attributes that have more influence over others in deciding whether a movie will be a hit or not. By analyzing various attributes of a movie, this project aims to achieve that.
The project consisted of Exploratory Data Analysis and Statistical Modelling of a movies database from Kaggle – "The Movies Dataset". This database consisted of attributes such as Cast, Crew, Plot keyword, Budget, Revenue, Posters, Release date, Languages, Production, Companies, Countries, IMDB votes, IMDB Ratings, and Vote average.

The project goal is achieved by the following 3 steps.

Importing and Tidying Data: The primary data table consisted of about 45k movies released before July 2017. Other data tables, specifically 'IMDB Ratings' and 'IMDB Movies' are used to source information of a few attributes into the primary data table to construct the final database that is used for performing further analysis. The data tables were not in tidy format and are parsed to create a tidy and usable database.

Exploratory Data Analysis: Univariate data visualization summaries are used to explore the relationship between the dependent and independent variables. These summaries are helpful to get insights into the correlation between any Movie attribute and how successful a movie is.

Feature Analysis: The influence of each attribute is estimated through Feature Importance Analysis using classification regression. Also, multiple regression models are run to compare performance of each model. The parameters chosen by the model can be viewed as decisive features. Following are the predictor variables: Release Year, Languages of Production, Popularity Score, IMDB Rating, IMDB Vote Count, Adult Rated (Y/N), Genre, Production Countries, Movie Run-time. The target variable is the ratio of Global Revenue and Budget, converted into a 1/0 to categorize a movie as Hit or Not Hit.

Data Sources

The Movie Dataset: This is the primary data set. It contains data of ~45k movies and has following columns:

Adult: A Boolean flag that tells whether a movie was 'Adult' rated or not
Belongs to collection: Tells the collection a movie belongs to (e.g., Lord of the Rings collection)
Budget: The budget of a movie. However, budget will be sourced from a different file as this column contains a lot of missing values
Genres: All the genres of a movie
Homepage: URL Link to the homepage of a movie
ID: Numeric ID of a movie
IMDB ID: Standard 9 Character IMDB ID of the format <ttxxxxxxx>

Original Language: Keys of Original Language of the movie in encoded as a key (e.g., 'en' for English)
Original Title: Original title of the movie in original language script
Overview: Summary about the story of the movie
Popularity: Continuous variable with a popularity score
Poster Path: JPG Image file name of the movie poster
Production Companies: Production house of a movie
Production Countries: Production countries of a movie
Release Date: Release date of a movie.
Revenue: Revenue earned. However, revenue will be sourced from a different file as this column contains a lot of missing values
Run-time: Run-time of a movie in minutes
Spoken Languages: All languages of the movie in {key:value} format
Status: Categorical variable showing status of the movie
Tagline: Tagline of a movie
Title: Title of the movie in English
Video: A Boolean Flag
Vote Average: IMDB rating of a movie. However, ratings data will be sourced from a different file as this column contains lot of discrepancies
Vote Count: #Ratings of a movie

Only the columns in bold are used from this file.

IMDB Movies dataset: This data set is a repository of ~85k movies from IMDB. It was used to source Budget, Revenue and Release Year.

Budget: Budget of a movie, not necessarily in USD
World-Wide Gross: Revenue earned by a movie globally
Year: Release Year of a movie

IMDB Ratings dataset: This data set is a repository of ~85k movies from IMDB. It is used to source only ratings of movies. The ratings data is continuous variable with range from 0-10

Methods:

Data Tidying

The data contained a lot of issues and needed to be cleaned before use. The following steps are used to prepare and tidy the data:

1. Merge main data, 'The Movie Dataset' with 'IMDB Movies Data' and 'IMDB Ratings Data' to source Budget, Revenue, Release Year and Ratings of movies
2. Select Relevant features
3. "Genres" and "Production Countries" data was present in a form of {key:value} pair and needed to be split. Further, each cell was populated with multiple {key:value} pairs, indicating that a movie has multiple genres and multiple production countries. The first step to tidy this data was to extract only

the *values* from the {key:value} pairs. This resulted in all genres/production countries being populated in a comma separated format. It was observed that overall, there were 20 unique genres and 111 production countries. Hence, 20 columns for each genre and 111 columns for each country were added to the dataset. Using the comma separated genres and countries in the first step, the 20 genre columns and 111 country columns were populated as 1 or 0. Thus, all genres and production countries a movie were obtained as separate features

4. Revenue and Budget Columns were populated with different currencies. Hence, they were converted to a single currency (USD) to bring the values on the same scale using the 'priceR' package. This package uses an API that sources exchange rates against USD of the current day from the world bank.
5. Y-Variable/Target variable: The Y-variable/Target variable of the model was created from the data. The ratio of the revenue and the budget was used to categorize a movie as 'Hit' or 'Not Hit'. The value of the ratio greater than 1 was categorized as 'Hit' and less than 1 was categorized as 'Not Hit'.

Exploratory Data Analysis:

Data scientists with the help of Exploratory data analysis (EDA) analyze and explore datasets and document important characteristics utilizing data visualization methods. It helps to determine how best to manipulate data sources to get the answers needed, and enables data scientists to discover patterns, identify anomalies, test hypotheses, or test assumptions. EDA is mainly used for studying what information can be inferenced other than formal modeling or hypothesis testing, which gives a better understanding of the variables in a data and the correlations between them. It can also help to determine if a particular statistical analysis method considered is suitable for analyzing the data. First developed by American mathematician John Tukey in the 1970s, the EDA method is still widely used in data retrieval processes.

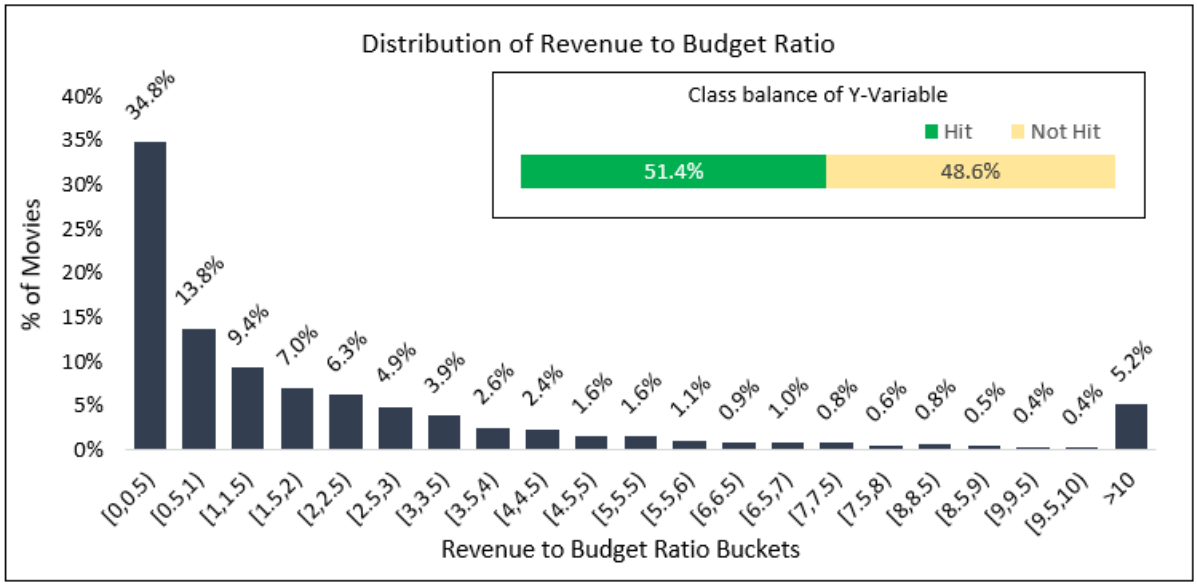


Figure 1: Distribution of Response Variable

Distribution of the Response Variable:

The response variable is calculated based on the ratio of already existing variables named revenue and budget. If the ratio of the revenue to budget comes out to be greater than 1, meaning profits were generated, it is marked as a hit movie, or is not a hit otherwise. The distribution of the response variable can be seen from figure 1. As expected, the number of movies decreases with increase in Revenue to Budget ratio. Also, 51.4% of movies have Revenue to Budget ratio greater than 1. Thus 51.4% of the movies are categorized as 'Hit' and remaining as 'non-Hit'. The Y-variable thus has a balanced class.

Relationship of variables with Y-Variable:

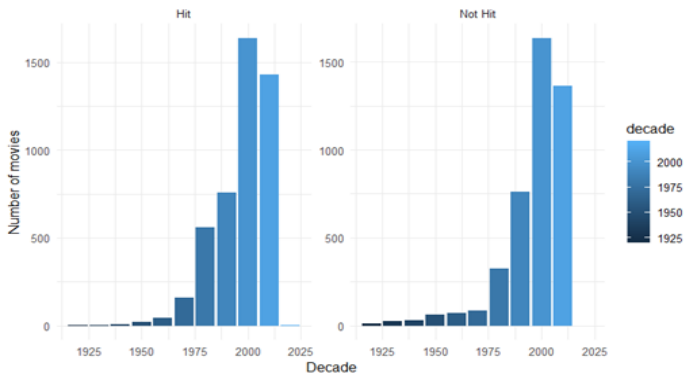


Figure 2: Time Series distribution of movies

Looking at *Figure 2*, we see that there is no strong correlation between the year a movie was released and whether it will be a hit or not. Another inference is that as time passes by, the number of hit and non-hit movies have increased. As expected, this variable does feature in the model

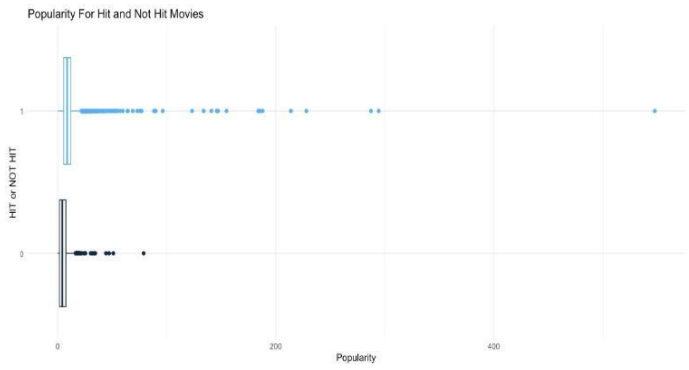


Figure 3: Distribution of Popularity Score

Looking at *Figure 3*, Popularity has a positive relationship with our target variable, Popularity is relatively more right skewed for hit movies. The median popularity is higher for hit movies, and both hit and non-hit movies are right skewed for Popularity. As expected, Popularity score features in the model

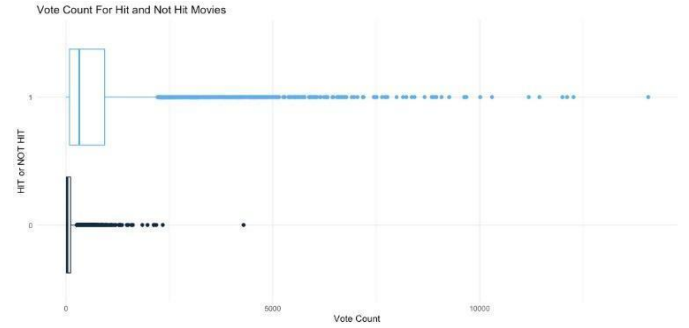


Figure 4: Distribution of Vote Count

Looking at *Figure 4*, Vote Count has a positive relationship with our target variable. The median Vote Count is higher for hit movies and both hit and not hit movies are right skewed for Vote Count. As expected, Vote Count features in the model

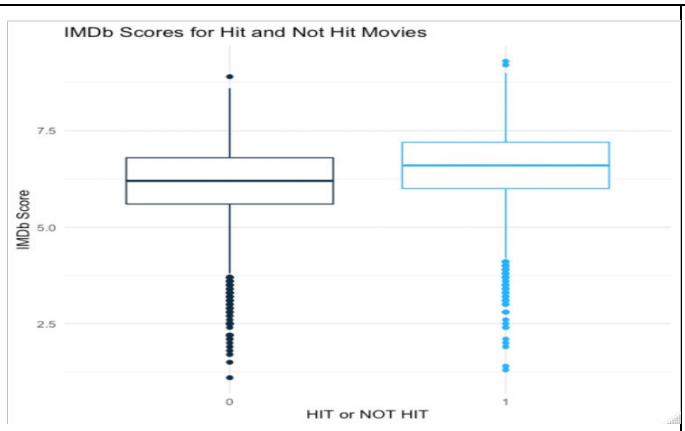


Figure 5: Distribution of IMDb Rating

Looking at *Figure 5*, Hit Movies have higher IMDB scores than non-Hit movies. As expected, IMDB Rating features in the model

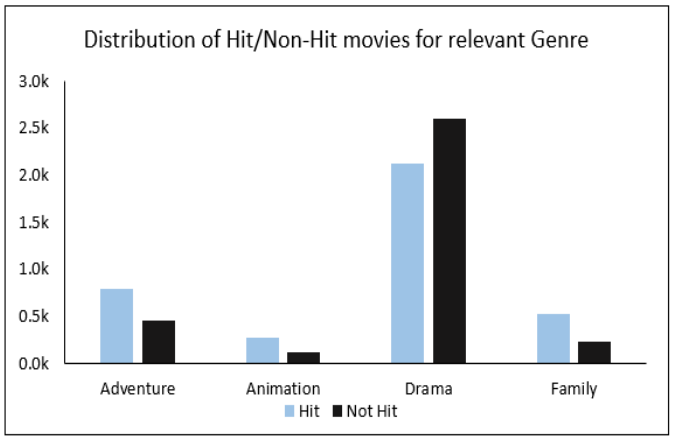


Figure 6: Distribution of Movies by Genre

From *Figure 6*, it can be inferred that Family, Animation and Adventure genres have higher percentage of hit movies which is also reflected in the correlation Matrix. On the other hand, Drama has a negative correlation which can be inferred from the adjacent figure. As expected, these four Genres feature in the model.

MODELLING:

Modelling is the methodology of trying to make a Machine Learning model to take in our dataset and predict a target variable. It also includes minimizing error, optimizing feature weights, and evaluating the model against different metrics such as Accuracy, Precision, Recall, F1-score, Area Under the Curve (AUC) etc. Four supervised machine learning models such as Logistic Regression, Decision Tree, KNN and Random Forest are chosen as models to train and test with our dataset.

LOGISTIC REGRESSION:

Logistic regression is a statistic that uses a logistic function to model a binary dependent variable. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a form of binary regression. Mathematically, a binary logistic model has a dependent variable with two possible values which is represented by an indicator variable, where the two values are labeled either 0 or 1.

K-NEAREST NEIGHBOURS (kNN):

The nearest neighbor (kNN) algorithm is a nonparametric classification method. Used for classification and regression. In either case, the input consists of the k closest training examples in the data set. Results depend on whether kNNs are used for classification or regression

DECISION TREE:

A decision tree is a flowchart-like structure in which each inner node represents a "test" for an attribute (such as whether a coin toss is a heads or tails), each branch represents a test result, and each leaf node represents a class. brand. (Determined after calculating all properties). The path from root to leaf is a classification rule.

RANDOM FOREST:

Random Forests or Random Decision Forests are ensemble training techniques for classification, regression, and other problems that work by constructing multiple decision trees during training. For classification problems, the output of a random forest is the class chosen from most trees. For regression problems, the mean or mean prediction of the individual trees is returned.

Choosing Model Predictors:

From the *Figure 7* below, there is strong correlation between surprising factors like Drama which has negative correlation with our target variable.

Movies from Countries like Belgium, Luxembourg has positive correlation with our target variable.

Family centered genres like Animation, Comedy, Family have strong correlation with our target feature.

A heat map of correlation coefficients describes the measure of relationship between variables on one axis with the variables on the other axis. According to the palette used here, the lighter the color the greater the relationship. The correlation heat map shown in the figure 7 shows the top features, based on correlation with the Y-variable. These features were use as the predictor variable.

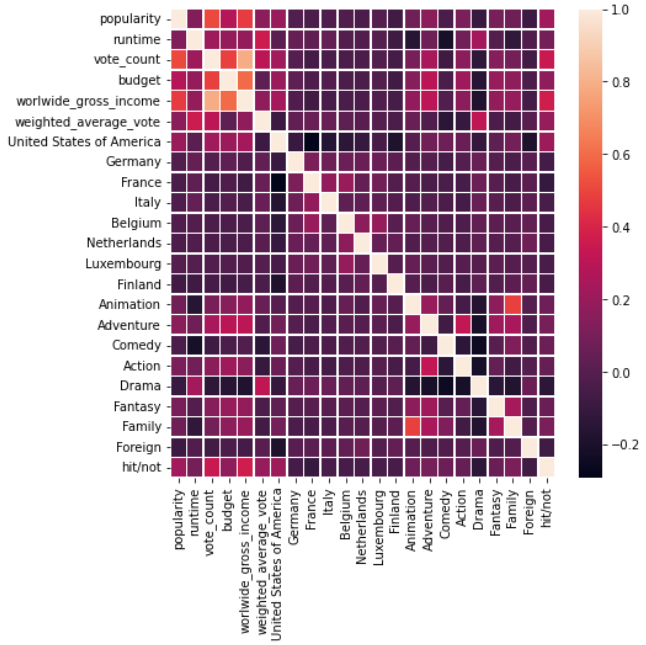


Figure 7: Correlation Heat Map for the Predictor Variables

Results:

Model	Specificity	Accuracy	Precision	F1-Score	Recall/ Sensitivity	AUC
Logistic Regression	1.0	100	1.0	1.0	1.0	1.0
kNN	0.9908	99.27	0.9913	0.9929	0.9945	0.9927
Decision Tree	0.8219	85.38	0.8395	0.8612	0.8840	0.8529
Random Forest	0.9527	95.27	0.9554	0.9538	0.9538	0.9527

Table 1: Model Evaluation Metrics

From Table 1 above:
A basic model such as Logistic regression and kNN performed better with our dataset when compared to complex models like decision Tree.
The reason Decision Tree has the worst, among the four, evaluation metrics might be due to presence of outliers in our training set.

Discussion:

Initial Data Analysis gave some interesting and unexpected relationships. Like how United States movies have more 'non-Hit' in comparison to other countries. This can be associated with the fact that the US has released a greater number of movies. Or how the genre "Drama" has negative correlation with whether a movie becomes a hit or not. It is also observed how Production Country has a strong correlation with our target variable. Features such as Production countries, specific genres, and the budget put into a movie also matters. Even though there are exceptions to each of the above-mentioned factors, in general, it is seen that a hit movie would have to be in one of the 8 genres.

The metrics (mentioned in *Table1*) shows the Evaluation metrics that determine the fit of models to our dataset. The modelling showed that the simplest model, Logistic Regression, with the simplest of error calculation methods give the best model. It is expected that models such as Random Forest and Decision Tree would perform better, but it is seen that decision tree has succumbed to outliers and exceptions and given the worst result out of the 4 models. Random Forest is not as sensitive to outliers, but it still did not out-perform Logistic Regression, even if hyperparameters were tuned.

Hence, movie makers, producers, Film distributors can make use of these above-mentioned factors to decide if the movie will be a hit or not. However, there may be exceptions to each case mentioned. This may be attributed to other important factors that make a movie hit or not, such as Story, screenplay, actors etc. In future, the analysis could include text mining on the summary of a movie. This could make the model even more accurate.

STATEMENT OF CONTRIBUTION:

Madhusudan Malhar Deshpande: Data Exploration and Data Tidying
Reem Ghabayen: Data Visualization and Documentation.
Arif Sarfaraz Waghbakriwala: Data Cleaning and Data Visualization.
Aanand Dhandapani: Data Cleaning and Data Modelling

REFERENCES:

Banik, R. (2017, November 09). The movies dataset. Kaggle. Retrieved November 16, 2021, from https://www.kaggle.com/rounakbanik/the-movies-dataset/code.

Leone, S. (2020, September 14). IMDB movies extensive dataset. Kaggle. Retrieved November 16, 2021, from https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset.

APPENDIX

```r
# Loading the csv file 'movies_metadata.csv' from the
# folder in the drive, and storing it into the variable
# 'stage_1'
library("tidyverse")
library("readr")
library("stringr")
library("dplyr")
library("priceR")
library("tibble")
library("gridExtra")
library("lemon")

setwd(getwd())
stage_1 <- as_tibble(read_csv("Movies_DataSet/movies_metadata.csv",
    show_col_types = FALSE))

png("1_raw_data_preview.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))

png("2_raw_data_summary.png", height = 50 * nrow(summary(stage_1)),
    width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
## # A tibble: 6 x 24
##   adult belongs_to_colle~  budget genres homepage    id imdb_id original_langua~
##   <lgl> <chr>               <dbl> <chr>  <chr>    <dbl> <chr>   <chr>
## 1 FALSE {'id': 10194, 'n~   3  e7 [{'id~ http://~   862 tt0114~ en
## 2 FALSE <NA>                6.5e7 [{'id~ <NA>      8844 tt0113~ en
## 3 FALSE {'id': 119050, '~   0     [{'id~ <NA>     15602 tt0113~ en
## 4 FALSE <NA>                1.6e7 [{'id~ <NA>     31357 tt0114~ en
## 5 FALSE {'id': 96871, 'n~   0     [{'id~ <NA>     11862 tt0113~ en
## 6 FALSE <NA>                6  e7 [{'id~ <NA>       949 tt0113~ en
## # ... with 16 more variables: original_title <chr>, overview <chr>,
## #   popularity <dbl>, poster_path <chr>, production_companies <chr>,
## #   production_countries <chr>, release_date <date>, revenue <dbl>,
## #   runtime <dbl>, spoken_languages <chr>, status <chr>, tagline <chr>,
## #   title <chr>, video <lgl>, vote_average <dbl>, vote_count <dbl>
```

```r
# Dividing the list of columns into the ones that are to be
# kept and the ones to be reserved
to_keep_columns <- c("adult", "genres", "imdb_id", "popularity",
    "runtime", "vote_count", "production_countries", "original_language",
    "title")
drop_columns <- c("belongs_to_collection", "homepage", "id",
    "budget", "poster_path", "video", "tagline", "production_companies",
    "overview", "release_date", "revenue", "status", "original_title",
    "vote_average")
```

```
### All the functions

# To check na values column wise
fun <- function(x) {
    tmp <- is.na(x)
    apply(tmp, 2, sum)
}

## Function to convert columns containing Dictionaries to
## List:
getAttribute <- function(vector) {
    vector <- as.vector(str_split(vector, regex("[\\[{':,}\\]]"))[[1]])
    vector <- vector[!vector == "" & !vector == " "]
    vector <- as.vector(vector[which(vector == "name") + 1])
    return(toString(vector))
}

## Converts Currecy as per today's curr value:
convert_currency <- function(datum) {
    # retrives a list of currencies seen in datum
    curr_type = unique(str_sub(datum, 1, 4))

    for (curr in curr_type) {
        # Fetches the currency Valye using priceR package
        exch_rate = exchange_rate_latest(curr)
        conversion_value = as.double(exch_rate[exch_rate[1] ==
            "USD"])[2]
        # Retrieved values in data with curr currency
        sub_datum = datum[str_sub(datum, 1, 4) == curr]
        for (data in sub_datum) {
            ind = which(datum == data)
            value = as.double(str_sub(data, 5))
            res = as.integer(value * conversion_value)
            datum[ind] = res
        }
    }
    return(datum)
}

split_cols <- function(x, colname, df) {

    ncols <- NULL
    colm <- NULL
    ncols <- max(stringr::str_count(x, ", ")) + 1
    colm <- paste(colname, 1:ncols, sep = "_")

    df <- tidyr::separate(data = df, col = colname, sep = ", ",
        into = colm, remove = FALSE)
    unique_val_list <- data.frame(matrix(ncol = 1, nrow = 0))
    colnames(unique_val_list) <- colm[1]
    for (i in colm) {
        colnames(unique_val_list) <- i
        tmp <- as.data.frame(unique(df[, i]))
```

```
        colnames(tmp) <- i
        unique_val_list <- rbind(as.data.frame(unique_val_list),
            tmp)
    }

    unique_val_list <- as.data.frame(unique(unique_val_list))
    unique_val_list <- as.data.frame(na.omit(unique_val_list))

    for (i in 1:length(unique_val_list[, 1])) {
        df[unique_val_list[i, 1]] <- 0
    }

    for (i in 1:nrow(df)) {
        for (j in colm) {
            if (!is.na(df[i, j])) {
                k <- as.character(df[i, j])
                df[i, k] = 1
            }
        }
    }
    df <- select(df, -colm)
    # filename <- paste(filename, '.csv')
    # write.csv(unique_val_list,filename, row.names =
    # FALSE)
    return(df)
}
```

```
## Keeping necessary Columns only.
stage_1 <- stage_1[to_keep_columns]

png("3_stage_1_columns_filtered.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))

png("4_stage_1_columns_filtered_summary.png", height = 50 * nrow(summary(stage_1)),
    width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
## # A tibble: 6 x 9
##   adult genres        imdb_id  popularity runtime vote_count production_countri~
##   <lgl> <chr>         <chr>         <dbl>   <dbl>      <dbl> <chr>
## 1 FALSE [{'id': 16, ~ tt01147~      21.9       81       5415 [{'iso_3166_1': 'U~
## 2 FALSE [{'id': 12, ~ tt01134~      17.0      104       2413 [{'iso_3166_1': 'U~
## 3 FALSE [{'id': 1074~ tt01132~      11.7      101         92 [{'iso_3166_1': 'U~
## 4 FALSE [{'id': 35, ~ tt01148~       3.86     127         34 [{'iso_3166_1': 'U~
## 5 FALSE [{'id': 35, ~ tt01130~       8.39     106        173 [{'iso_3166_1': 'U~
## 6 FALSE [{'id': 28, ~ tt01132~      17.9      170       1886 [{'iso_3166_1': 'U~
## # ... with 2 more variables: original_language <chr>, title <chr>
```

```
## Converting all Dictionary kinda Cols into Lists
stage_1$genres <- sapply(stage_1$genres, getAttribute, USE.NAMES = FALSE,
    simplify = "array")  # Genres Column
stage_1$production_countries <- sapply(stage_1$production_countries,
    getAttribute, USE.NAMES = FALSE, simplify = "array")

png("5_stage_1_post_dict_str_conversion.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))

png("6_stage_1_post_dict_str_conversion_summary.png", height = 50 *
    nrow(summary(stage_1)), width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
## # A tibble: 6 x 9
##   adult genres         imdb_id  popularity runtime vote_count production_countr~
##   <lgl> <chr>          <chr>         <dbl>   <dbl>      <dbl> <chr>
## 1 FALSE Animation, Co~ tt01147~      21.9       81       5415 United States of ~
## 2 FALSE Adventure, Fa~ tt01134~      17.0      104       2413 United States of ~
## 3 FALSE Romance, Come~ tt01132~      11.7      101         92 United States of ~
## 4 FALSE Comedy, Drama~ tt01148~       3.86     127         34 United States of ~
## 5 FALSE Comedy         tt01130~       8.39     106        173 United States of ~
## 6 FALSE Action, Crime~ tt01132~      17.9      170       1886 United States of ~
## # ... with 2 more variables: original_language <chr>, title <chr>
```

```
# Replacing blank values with NA and then omitting the NAs.
stage_1 <- stage_1 %>%
    mutate(genres = ifelse(genres == "", NA, genres)) %>%
    mutate(production_countries = ifelse(production_countries ==
        "", NA, production_countries))

png("7_stage_1_post_blank_val_removal.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))

png("8_stage_1_post_blank_val_removal_summary.png", height = 50 *
    nrow(summary(stage_1)), width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
## # A tibble: 6 x 9
##   adult genres         imdb_id  popularity runtime vote_count production_countr~
##   <lgl> <chr>          <chr>         <dbl>   <dbl>      <dbl> <chr>
## 1 FALSE Animation, Co~ tt01147~      21.9       81       5415 United States of ~
## 2 FALSE Adventure, Fa~ tt01134~      17.0      104       2413 United States of ~
## 3 FALSE Romance, Come~ tt01132~      11.7      101         92 United States of ~
## 4 FALSE Comedy, Drama~ tt01148~       3.86     127         34 United States of ~
## 5 FALSE Comedy         tt01130~       8.39     106        173 United States of ~
## 6 FALSE Action, Crime~ tt01132~      17.9      170       1886 United States of ~
## # ... with 2 more variables: original_language <chr>, title <chr>
```

```r
# Joining the files movies metadata and IMDB movies.
IMDB_movies <- as_tibble(read_csv("IMDb movies.csv", show_col_types = FALSE))
IMDB_rating <- as_tibble(read_csv("IMDb ratings.csv", show_col_types = FALSE))
stage_1 <- dplyr::inner_join(stage_1, select(IMDB_movies, year,
    imdb_title_id, director, budget, worlwide_gross_income),
    by = c(imdb_id = "imdb_title_id"))
stage_1 <- dplyr::inner_join(stage_1, select(IMDB_rating, imdb_title_id,
    weighted_average_vote), by = c(imdb_id = "imdb_title_id"))
stage_1 <- na.omit(stage_1)

png("9_stage_1_post_merge.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))

png("10_stage_1_post_merge.png", height = 50 * nrow(summary(stage_1)),
    width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
## # A tibble: 6 x 14
##   adult genres         imdb_id  popularity runtime vote_count production_countr~
##   <lgl> <chr>          <chr>         <dbl>   <dbl>      <dbl> <chr>
## 1 FALSE Animation, Co~ tt01147~      21.9       81       5415 United States of ~
## 2 FALSE Adventure, Fa~ tt01134~      17.0      104       2413 United States of ~
## 3 FALSE Romance, Come~ tt01132~      11.7      101         92 United States of ~
## 4 FALSE Comedy, Drama~ tt01148~       3.86     127         34 United States of ~
## 5 FALSE Comedy         tt01130~       8.39     106        173 United States of ~
## 6 FALSE Action, Crime~ tt01132~      17.9      170       1886 United States of ~
## # ... with 7 more variables: original_language <chr>, title <chr>, year <dbl>,
## #   director <chr>, budget <chr>, worlwide_gross_income <chr>,
## #   weighted_average_vote <dbl>
```

```r
# Currency Conversion and Dollar Removal
stage_1$budget[!str_detect(stage_1$budget, "^\\$")] = convert_currency(stage_1$budget[!str_detect(stage_
    "^\\$")])  # Currency Conversion
```

```
## Daily GBP  exchange rate as at end of day 2021-12-14 GMT
## Daily EUR  exchange rate as at end of day 2021-12-13 GMT
## Daily CAD  exchange rate as at end of day 2021-12-13 GMT
## Daily FRF  exchange rate as at end of day 2021-12-14 GMT
## Daily DEM  exchange rate as at end of day 2021-12-14 GMT
## Daily AUD  exchange rate as at end of day 2021-12-14 GMT
## Daily DKK  exchange rate as at end of day 2021-12-14 GMT
## Daily JPY  exchange rate as at end of day 2021-12-14 GMT
## Daily HKD  exchange rate as at end of day 2021-12-14 GMT
## Daily RUR  exchange rate as at end of day 2021-12-14 GMT
## Daily ITL  exchange rate as at end of day 2021-12-14 GMT
## Daily ESP  exchange rate as at end of day 2021-12-13 GMT
## Daily BEF  exchange rate as at end of day 2021-12-14 GMT
## Daily SEK  exchange rate as at end of day 2021-12-14 GMT
## Daily INR  exchange rate as at end of day 2021-12-13 GMT
## Daily IEP  exchange rate as at end of day 2021-12-14 GMT
```

```
## Daily ATS  exchange rate as at end of day 2021-12-14 GMT
## Daily NOK  exchange rate as at end of day 2021-12-14 GMT
## Daily BRL  exchange rate as at end of day 2021-12-14 GMT
## Daily FIM  exchange rate as at end of day 2021-12-14 GMT
## Daily SGD  exchange rate as at end of day 2021-12-14 GMT
## Daily THB  exchange rate as at end of day 2021-12-14 GMT
## Daily NLG  exchange rate as at end of day 2021-12-14 GMT
## Daily CNY  exchange rate as at end of day 2021-12-14 GMT
## Daily HUF  exchange rate as at end of day 2021-12-14 GMT
## Daily CZK  exchange rate as at end of day 2021-12-14 GMT
## Daily PLN  exchange rate as at end of day 2021-12-14 GMT
## Daily KRW  exchange rate as at end of day 2021-12-13 GMT
## Daily CHF  exchange rate as at end of day 2021-12-14 GMT
## Daily ISK  exchange rate as at end of day 2021-12-14 GMT
## Daily EGP  exchange rate as at end of day 2021-12-14 GMT
## Daily BGL  exchange rate as at end of day 2021-12-14 GMT
## Daily TWD  exchange rate as at end of day 2021-12-14 GMT
## Daily MXN  exchange rate as at end of day 2021-12-14 GMT
## Daily LTL  exchange rate as at end of day 2021-12-13 GMT
## Daily NZD  exchange rate as at end of day 2021-12-14 GMT
## Daily ARS  exchange rate as at end of day 2021-12-14 GMT
## Daily VEB  exchange rate as at end of day 2021-12-14 GMT
## Daily NGN  exchange rate as at end of day 2021-12-14 GMT
## Daily LVL  exchange rate as at end of day 2021-12-14 GMT
## Daily ZAR  exchange rate as at end of day 2021-12-14 GMT
## Daily PKR  exchange rate as at end of day 2021-12-14 GMT
## Daily TRL  exchange rate as at end of day 2021-12-14 GMT
## Daily IDR  exchange rate as at end of day 2021-12-14 GMT
## Daily PHP  exchange rate as at end of day 2021-12-14 GMT
## Daily ILS  exchange rate as at end of day 2021-12-14 GMT
## Daily AMD  exchange rate as at end of day 2021-12-14 GMT
```

```
stage_1$worlwide_gross_income[!str_detect(stage_1$worlwide_gross_income,
    "^\\$")] = convert_currency(stage_1$worlwide_gross_income[!str_detect(stage_1$worlwide_gross_income
    "^\\$")])  # Currency Conversion
stage_1 = na.omit(stage_1)

stage_1$budget[str_detect(stage_1$budget, "^\\$")] = as.numeric(str_sub(stage_1$budget[str_detect(stage
    "^\\$")], 3))  # Dollar removal
stage_1$worlwide_gross_income[str_detect(stage_1$worlwide_gross_income,
    "^\\$")] = as.numeric(str_sub(stage_1$worlwide_gross_income[str_detect(stage_1$worlwide_gross_incom
    "^\\$")], 3))  #Dollar Removal
stage_1$budget = as.numeric(stage_1$budget)
stage_1$worlwide_gross_income = as.numeric(stage_1$worlwide_gross_income)

stage_1 = stage_1 %>%
    mutate(`hit/not` = ifelse(worlwide_gross_income/budget >
        1, 1, 0))

stage_1 <- na.omit(stage_1)

png("11_stage_1_post_cc.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))
```

```
png("12_stage_1_post_cc.png", height = 50 * nrow(summary(stage_1)),
    width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
## # A tibble: 6 x 15
##   adult genres         imdb_id  popularity runtime vote_count production_countr~
##   <lgl> <chr>          <chr>         <dbl>   <dbl>      <dbl> <chr>
## 1 FALSE Animation, Co~ tt01147~      21.9       81       5415 United States of ~
## 2 FALSE Adventure, Fa~ tt01134~      17.0      104       2413 United States of ~
## 3 FALSE Romance, Come~ tt01132~      11.7      101         92 United States of ~
## 4 FALSE Comedy, Drama~ tt01148~       3.86     127         34 United States of ~
## 5 FALSE Comedy         tt01130~       8.39     106        173 United States of ~
## 6 FALSE Action, Crime~ tt01132~      17.9      170       1886 United States of ~
## # ... with 8 more variables: original_language <chr>, title <chr>, year <dbl>,
## #   director <chr>, budget <dbl>, worlwide_gross_income <dbl>,
## #   weighted_average_vote <dbl>, hit/not <dbl>
```

```
## Calling the split_cols function to convert production
## companies into columns and sparse filling the cells
stage_1 <- as.data.frame(split_cols(stage_1$production_countries,
    "production_countries", stage_1))

## Calling the split_cols function to convert genres into
## columns and sparse filling the cells
stage_1 <- as.data.frame(split_cols(stage_1$genres, "genres",
    stage_1))

png("13_stage_1_post_pivoting.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))

png("14_stage_1_post_pivoting.png", height = 50 * nrow(summary(stage_1)),
    width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
##   adult                          genres   imdb_id popularity runtime vote_count
## 1 FALSE      Animation, Comedy, Family tt0114709  21.946943      81       5415
## 2 FALSE     Adventure, Fantasy, Family tt0113497  17.015539     104       2413
## 3 FALSE                Romance, Comedy tt0113228  11.712900     101         92
## 4 FALSE         Comedy, Drama, Romance tt0114885   3.859495     127         34
## 5 FALSE                         Comedy tt0113041   8.387519     106        173
## 6 FALSE Action, Crime, Drama, Thriller tt0113277  17.924927     170       1886
##       production_countries original_language                     title year
## 1 United States of America                en                 Toy Story 1995
## 2 United States of America                en                   Jumanji 1995
## 3 United States of America                en          Grumpier Old Men 1995
## 4 United States of America                en         Waiting to Exhale 1995
## 5 United States of America                en Father of the Bride Part II 1995
```

```
## 6 United States of America                en                        Heat 1995
##          director  budget worlwide_gross_income weighted_average_vote hit/not
## 1   John Lasseter 3.0e+07             404265438                   8.3       1
## 2    Joe Johnston 6.5e+07             262821940                   7.0       1
## 3   Howard Deutch 2.5e+07              71518503                   6.7       1
## 4 Forest Whitaker 1.6e+07              81452156                   5.9       1
## 5   Charles Shyer 3.0e+07              76594107                   6.1       1
## 6    Michael Mann 6.0e+07             187436818                   8.2       1
##   United States of America Germany United Kingdom France Italy Australia
## 1                        1       0              0      0     0         0
## 2                        1       0              0      0     0         0
## 3                        1       0              0      0     0         0
## 4                        1       0              0      0     0         0
## 5                        1       0              0      0     0         0
## 6                        1       0              0      0     0         0
##   Belgium Canada Iran Netherlands Hong Kong Japan Austria New Zealand Mexico
## 1       0      0    0           0         0     0       0           0      0
## 2       0      0    0           0         0     0       0           0      0
## 3       0      0    0           0         0     0       0           0      0
## 4       0      0    0           0         0     0       0           0      0
## 5       0      0    0           0         0     0       0           0      0
## 6       0      0    0           0         0     0       0           0      0
##   Taiwan Peru China South Africa Denmark Spain Serbia Sweden Czech Republic
## 1      0    0     0            0       0     0      0      0              0
## 2      0    0     0            0       0     0      0      0              0
## 3      0    0     0            0       0     0      0      0              0
## 4      0    0     0            0       0     0      0      0              0
## 5      0    0     0            0       0     0      0      0              0
## 6      0    0     0            0       0     0      0      0              0
##   Ireland Trinidad and Tobago Russia India Brazil Aruba Israel Luxembourg
## 1       0                   0      0     0      0     0      0           0
## 2       0                   0      0     0      0     0      0           0
## 3       0                   0      0     0      0     0      0           0
## 4       0                   0      0     0      0     0      0           0
## 5       0                   0      0     0      0     0      0           0
## 6       0                   0      0     0      0     0      0           0
##   Argentina Ecuador Bahamas Malaysia Switzerland Bulgaria Thailand Namibia
## 1         0       0       0        0           0        0        0       0
## 2         0       0       0        0           0        0        0       0
## 3         0       0       0        0           0        0        0       0
## 4         0       0       0        0           0        0        0       0
## 5         0       0       0        0           0        0        0       0
## 6         0       0       0        0           0        0        0       0
##   South Korea Norway Finland Afghanistan Iceland Romania Soviet Union Hungary
## 1           0      0       0           0       0       0            0       0
## 2           0      0       0           0       0       0            0       0
## 3           0      0       0           0       0       0            0       0
## 4           0      0       0           0       0       0            0       0
## 5           0      0       0           0       0       0            0       0
## 6           0      0       0           0       0       0            0       0
##   Chile Bhutan Poland Palestinian Territory Uruguay Turkey Morocco Algeria
## 1     0      0      0                     0       0      0       0       0
## 2     0      0      0                     0       0      0       0       0
## 3     0      0      0                     0       0      0       0       0
```

```
## 4     0      0      0                    0       0      0       0       0
## 5     0      0      0                    0       0      0       0       0
## 6     0      0      0                    0       0      0       0       0
##   Singapore Mongolia Bosnia and Herzegovina Mali Lebanon Kazakhstan Greece
## 1         0        0                    0    0       0          0      0
## 2         0        0                    0    0       0          0      0
## 3         0        0                    0    0       0          0      0
## 4         0        0                    0    0       0          0      0
## 5         0        0                    0    0       0          0      0
## 6         0        0                    0    0       0          0      0
##   United Arab Emirates Indonesia Egypt Slovenia Macedonia Estonia Portugal
## 1                    0         0     0        0         0       0        0
## 2                    0         0     0        0         0       0        0
## 3                    0         0     0        0         0       0        0
## 4                    0         0     0        0         0       0        0
## 5                    0         0     0        0         0       0        0
## 6                    0         0     0        0         0       0        0
##   Mauritania Cyprus Bangladesh Vietnam Lithuania Jordan Nigeria Philippines
## 1          0      0          0       0         0      0       0           0
## 2          0      0          0       0         0      0       0           0
## 3          0      0          0       0         0      0       0           0
## 4          0      0          0       0         0      0       0           0
## 5          0      0          0       0         0      0       0           0
## 6          0      0          0       0         0      0       0           0
##   Venezuela Pakistan Burkina Faso Latvia Cuba Malta Qatar Samoa Ukraine
## 1         0        0           0      0    0     0     0     0       0
## 2         0        0           0      0    0     0     0     0       0
## 3         0        0           0      0    0     0     0     0       0
## 4         0        0           0      0    0     0     0     0       0
## 5         0        0           0      0    0     0     0     0       0
## 6         0        0           0      0    0     0     0     0       0
##   Colombia Cambodia Panama Georgia Dominican Republic Azerbaijan Armenia
## 1        0        0      0       0                  0          0       0
## 2        0        0      0       0                  0          0       0
## 3        0        0      0       0                  0          0       0
## 4        0        0      0       0                  0          0       0
## 5        0        0      0       0                  0          0       0
## 6        0        0      0       0                  0          0       0
##   Botswana Croatia Costa Rica Ghana Tunisia Rwanda Angola Monaco Puerto Rico
## 1        0       0          0     0       0      0      0      0           0
## 2        0       0          0     0       0      0      0      0           0
## 3        0       0          0     0       0      0      0      0           0
## 4        0       0          0     0       0      0      0      0           0
## 5        0       0          0     0       0      0      0      0           0
## 6        0       0          0     0       0      0      0      0           0
##    "Lao People Slovakia Gibraltar Liechtenstein Chad Iraq Serbia and Montenegro
## 1            0        0         0             0    0    0                     0
## 2            0        0         0             0    0    0                     0
## 3            0        0         0             0    0    0                     0
## 4            0        0         0             0    0    0                     0
## 5            0        0         0             0    0    0                     0
## 6            0        0         0             0    0    0                     0
##   Paraguay Animation Adventure Romance Comedy Action History Drama Crime
## 1        0         1         0       0      1      0       0     0     0
```

```
## 2        0         0         1       0      0      0       0     0     0
## 3        0         0         0       1      1      0       0     0     0
## 4        0         0         0       1      1      0       0     1     0
## 5        0         0         0       0      1      0       0     0     0
## 6        0         0         0       0      0      1       0     1     1
##   Fantasy Science Fiction Music Horror Family Mystery Thriller Western War
## 1       0               0     0      0      1       0        0       0   0
## 2       1               0     0      0      1       0        0       0   0
## 3       0               0     0      0      0       0        0       0   0
## 4       0               0     0      0      0       0        0       0   0
## 5       0               0     0      0      0       0        0       0   0
## 6       0               0     0      0      0       0        1       0   0
##   Documentary TV Movie Foreign
## 1           0        0       0
## 2           0        0       0
## 3           0        0       0
## 4           0        0       0
## 5           0        0       0
## 6           0        0       0
```

```
## Converting the abbreviations into full forms for
## langauage column
lang_codes <- as_tibble(read_csv("language_codes_csv.csv", show_col_types = FALSE))
stage_1 <- dplyr::left_join(stage_1, lang_codes, by = c(original_language = "alpha2"),
    keep = FALSE)

png("15_stage_1_post_lang_codes.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))

png("16_stage_1_post_lang_codes.png", height = 50 * nrow(summary(stage_1)),
    width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
##   adult                          genres   imdb_id popularity runtime vote_count
## 1 FALSE      Animation, Comedy, Family tt0114709  21.946943      81       5415
## 2 FALSE     Adventure, Fantasy, Family tt0113497  17.015539     104       2413
## 3 FALSE                Romance, Comedy tt0113228  11.712900     101         92
## 4 FALSE         Comedy, Drama, Romance tt0114885   3.859495     127         34
## 5 FALSE                         Comedy tt0113041   8.387519     106        173
## 6 FALSE Action, Crime, Drama, Thriller tt0113277  17.924927     170       1886
##       production_countries original_language                     title year
## 1 United States of America                en                 Toy Story 1995
## 2 United States of America                en                   Jumanji 1995
## 3 United States of America                en          Grumpier Old Men 1995
## 4 United States of America                en         Waiting to Exhale 1995
## 5 United States of America                en Father of the Bride Part II 1995
## 6 United States of America                en                      Heat 1995
##          director  budget worlwide_gross_income weighted_average_vote hit/not
## 1   John Lasseter 3.0e+07            404265438                   8.3       1
## 2    Joe Johnston 6.5e+07            262821940                   7.0       1
## 3   Howard Deutch 2.5e+07             71518503                   6.7       1
```

```
## 4 Forest Whitaker 1.6e+07              81452156                  5.9       1
## 5   Charles Shyer 3.0e+07              76594107                  6.1       1
## 6    Michael Mann 6.0e+07             187436818                  8.2       1
##   United States of America Germany United Kingdom France Italy Australia
## 1                        1       0              0      0     0         0
## 2                        1       0              0      0     0         0
## 3                        1       0              0      0     0         0
## 4                        1       0              0      0     0         0
## 5                        1       0              0      0     0         0
## 6                        1       0              0      0     0         0
##   Belgium Canada Iran Netherlands Hong Kong Japan Austria New Zealand Mexico
## 1       0      0    0           0         0     0       0           0      0
## 2       0      0    0           0         0     0       0           0      0
## 3       0      0    0           0         0     0       0           0      0
## 4       0      0    0           0         0     0       0           0      0
## 5       0      0    0           0         0     0       0           0      0
## 6       0      0    0           0         0     0       0           0      0
##   Taiwan Peru China South Africa Denmark Spain Serbia Sweden Czech Republic
## 1      0    0     0            0       0     0      0      0              0
## 2      0    0     0            0       0     0      0      0              0
## 3      0    0     0            0       0     0      0      0              0
## 4      0    0     0            0       0     0      0      0              0
## 5      0    0     0            0       0     0      0      0              0
## 6      0    0     0            0       0     0      0      0              0
##   Ireland Trinidad and Tobago Russia India Brazil Aruba Israel Luxembourg
## 1       0                   0      0     0      0     0      0          0
## 2       0                   0      0     0      0     0      0          0
## 3       0                   0      0     0      0     0      0          0
## 4       0                   0      0     0      0     0      0          0
## 5       0                   0      0     0      0     0      0          0
## 6       0                   0      0     0      0     0      0          0
##   Argentina Ecuador Bahamas Malaysia Switzerland Bulgaria Thailand Namibia
## 1         0       0       0        0           0        0        0       0
## 2         0       0       0        0           0        0        0       0
## 3         0       0       0        0           0        0        0       0
## 4         0       0       0        0           0        0        0       0
## 5         0       0       0        0           0        0        0       0
## 6         0       0       0        0           0        0        0       0
##   South Korea Norway Finland Afghanistan Iceland Romania Soviet Union Hungary
## 1           0      0       0           0       0       0            0       0
## 2           0      0       0           0       0       0            0       0
## 3           0      0       0           0       0       0            0       0
## 4           0      0       0           0       0       0            0       0
## 5           0      0       0           0       0       0            0       0
## 6           0      0       0           0       0       0            0       0
##   Chile Bhutan Poland Palestinian Territory Uruguay Turkey Morocco Algeria
## 1     0      0      0                     0       0      0       0       0
## 2     0      0      0                     0       0      0       0       0
## 3     0      0      0                     0       0      0       0       0
## 4     0      0      0                     0       0      0       0       0
## 5     0      0      0                     0       0      0       0       0
## 6     0      0      0                     0       0      0       0       0
##   Singapore Mongolia Bosnia and Herzegovina Mali Lebanon Kazakhstan Greece
## 1         0        0                     0    0       0          0      0
```

```
## 2         0        0                    0    0       0          0      0
## 3         0        0                    0    0       0          0      0
## 4         0        0                    0    0       0          0      0
## 5         0        0                    0    0       0          0      0
## 6         0        0                    0    0       0          0      0
##   United Arab Emirates Indonesia Egypt Slovenia Macedonia Estonia Portugal
## 1                    0         0     0        0         0       0        0
## 2                    0         0     0        0         0       0        0
## 3                    0         0     0        0         0       0        0
## 4                    0         0     0        0         0       0        0
## 5                    0         0     0        0         0       0        0
## 6                    0         0     0        0         0       0        0
##   Mauritania Cyprus Bangladesh Vietnam Lithuania Jordan Nigeria Philippines
## 1          0      0          0       0         0      0       0           0
## 2          0      0          0       0         0      0       0           0
## 3          0      0          0       0         0      0       0           0
## 4          0      0          0       0         0      0       0           0
## 5          0      0          0       0         0      0       0           0
## 6          0      0          0       0         0      0       0           0
##   Venezuela Pakistan Burkina Faso Latvia Cuba Malta Qatar Samoa Ukraine
## 1         0        0            0      0    0     0     0     0       0
## 2         0        0            0      0    0     0     0     0       0
## 3         0        0            0      0    0     0     0     0       0
## 4         0        0            0      0    0     0     0     0       0
## 5         0        0            0      0    0     0     0     0       0
## 6         0        0            0      0    0     0     0     0       0
##   Colombia Cambodia Panama Georgia Dominican Republic Azerbaijan Armenia
## 1        0        0      0       0                  0          0       0
## 2        0        0      0       0                  0          0       0
## 3        0        0      0       0                  0          0       0
## 4        0        0      0       0                  0          0       0
## 5        0        0      0       0                  0          0       0
## 6        0        0      0       0                  0          0       0
##   Botswana Croatia Costa Rica Ghana Tunisia Rwanda Angola Monaco Puerto Rico
## 1        0       0          0     0       0      0      0      0           0
## 2        0       0          0     0       0      0      0      0           0
## 3        0       0          0     0       0      0      0      0           0
## 4        0       0          0     0       0      0      0      0           0
## 5        0       0          0     0       0      0      0      0           0
## 6        0       0          0     0       0      0      0      0           0
##    "Lao People Slovakia Gibraltar Liechtenstein Chad Iraq Serbia and Montenegro
## 1            0        0         0             0    0    0                     0
## 2            0        0         0             0    0    0                     0
## 3            0        0         0             0    0    0                     0
## 4            0        0         0             0    0    0                     0
## 5            0        0         0             0    0    0                     0
## 6            0        0         0             0    0    0                     0
##   Paraguay Animation Adventure Romance Comedy Action History Drama Crime
## 1        0         1         0       0      1      0       0     0     0
## 2        0         0         1       0      0      0       0     0     0
## 3        0         0         0       1      1      0       0     0     0
## 4        0         0         0       1      1      0       0     1     0
## 5        0         0         0       0      1      0       0     0     0
## 6        0         0         0       0      0      1       0     1     1
```

```
##   Fantasy Science Fiction Music Horror Family Mystery Thriller Western War
## 1       0               0     0      0      1       0        0       0   0
## 2       1               0     0      0      1       0        0       0   0
## 3       0               0     0      0      0       0        0       0   0
## 4       0               0     0      0      0       0        0       0   0
## 5       0               0     0      0      0       0        0       0   0
## 6       0               0     0      0      0       0        1       0   0
##   Documentary TV Movie Foreign English
## 1           0        0       0 English
## 2           0        0       0 English
## 3           0        0       0 English
## 4           0        0       0 English
## 5           0        0       0 English
## 6           0        0       0 English
```

```
## relocating the response variable to the last position
## column wise
stage_1 <- relocate(stage_1, `hit/not`, .after = last_col())

png("17_stage_1_post_relocation_n_final.png", height = 50 * nrow(head(stage_1)),
    width = 200 * ncol(head(stage_1)))
grid.table(head(stage_1))

png("18_stage_1_post_relocation_n_final", height = 50 * nrow(summary(stage_1)),
    width = 150 * ncol(summary(stage_1)))
grid.table(summary(stage_1))

head(stage_1)
```

```
##   adult                          genres   imdb_id popularity runtime vote_count
## 1 FALSE      Animation, Comedy, Family tt0114709  21.946943      81       5415
## 2 FALSE     Adventure, Fantasy, Family tt0113497  17.015539     104       2413
## 3 FALSE                Romance, Comedy tt0113228  11.712900     101         92
## 4 FALSE         Comedy, Drama, Romance tt0114885   3.859495     127         34
## 5 FALSE                         Comedy tt0113041   8.387519     106        173
## 6 FALSE Action, Crime, Drama, Thriller tt0113277  17.924927     170       1886
##       production_countries original_language                     title year
## 1 United States of America               en                 Toy Story 1995
## 2 United States of America               en                   Jumanji 1995
## 3 United States of America               en          Grumpier Old Men 1995
## 4 United States of America               en         Waiting to Exhale 1995
## 5 United States of America               en Father of the Bride Part II 1995
## 6 United States of America               en                      Heat 1995
##          director  budget worlwide_gross_income weighted_average_vote
## 1   John Lasseter 3.0e+07             404265438                   8.3
## 2    Joe Johnston 6.5e+07             262821940                   7.0
## 3   Howard Deutch 2.5e+07              71518503                   6.7
## 4 Forest Whitaker 1.6e+07              81452156                   5.9
## 5   Charles Shyer 3.0e+07              76594107                   6.1
## 6    Michael Mann 6.0e+07             187436818                   8.2
##   United States of America Germany United Kingdom France Italy Australia
## 1                        1       0              0      0     0         0
## 2                        1       0              0      0     0         0
## 3                        1       0              0      0     0         0
```

```
## 4                      1       0             0      0     0         0
## 5                      1       0             0      0     0         0
## 6                      1       0             0      0     0         0
##   Belgium Canada Iran Netherlands Hong Kong Japan Austria New Zealand Mexico
## 1       0      0    0           0         0     0       0           0      0
## 2       0      0    0           0         0     0       0           0      0
## 3       0      0    0           0         0     0       0           0      0
## 4       0      0    0           0         0     0       0           0      0
## 5       0      0    0           0         0     0       0           0      0
## 6       0      0    0           0         0     0       0           0      0
##   Taiwan Peru China South Africa Denmark Spain Serbia Sweden Czech Republic
## 1      0    0     0            0       0     0      0      0              0
## 2      0    0     0            0       0     0      0      0              0
## 3      0    0     0            0       0     0      0      0              0
## 4      0    0     0            0       0     0      0      0              0
## 5      0    0     0            0       0     0      0      0              0
## 6      0    0     0            0       0     0      0      0              0
##   Ireland Trinidad and Tobago Russia India Brazil Aruba Israel Luxembourg
## 1       0                   0      0     0      0     0      0          0
## 2       0                   0      0     0      0     0      0          0
## 3       0                   0      0     0      0     0      0          0
## 4       0                   0      0     0      0     0      0          0
## 5       0                   0      0     0      0     0      0          0
## 6       0                   0      0     0      0     0      0          0
##   Argentina Ecuador Bahamas Malaysia Switzerland Bulgaria Thailand Namibia
## 1         0       0       0        0           0        0        0       0
## 2         0       0       0        0           0        0        0       0
## 3         0       0       0        0           0        0        0       0
## 4         0       0       0        0           0        0        0       0
## 5         0       0       0        0           0        0        0       0
## 6         0       0       0        0           0        0        0       0
##   South Korea Norway Finland Afghanistan Iceland Romania Soviet Union Hungary
## 1           0      0       0           0       0       0            0       0
## 2           0      0       0           0       0       0            0       0
## 3           0      0       0           0       0       0            0       0
## 4           0      0       0           0       0       0            0       0
## 5           0      0       0           0       0       0            0       0
## 6           0      0       0           0       0       0            0       0
##   Chile Bhutan Poland Palestinian Territory Uruguay Turkey Morocco Algeria
## 1     0      0      0                     0       0      0       0       0
## 2     0      0      0                     0       0      0       0       0
## 3     0      0      0                     0       0      0       0       0
## 4     0      0      0                     0       0      0       0       0
## 5     0      0      0                     0       0      0       0       0
## 6     0      0      0                     0       0      0       0       0
##   Singapore Mongolia Bosnia and Herzegovina Mali Lebanon Kazakhstan Greece
## 1         0        0                     0    0       0          0      0
## 2         0        0                     0    0       0          0      0
## 3         0        0                     0    0       0          0      0
## 4         0        0                     0    0       0          0      0
## 5         0        0                     0    0       0          0      0
## 6         0        0                     0    0       0          0      0
##   United Arab Emirates Indonesia Egypt Slovenia Macedonia Estonia Portugal
## 1                    0         0     0        0         0       0        0
```

```
## 2                   0         0     0        0         0       0        0
## 3                   0         0     0        0         0       0        0
## 4                   0         0     0        0         0       0        0
## 5                   0         0     0        0         0       0        0
## 6                   0         0     0        0         0       0        0
##   Mauritania Cyprus Bangladesh Vietnam Lithuania Jordan Nigeria Philippines
## 1          0      0          0       0         0      0       0           0
## 2          0      0          0       0         0      0       0           0
## 3          0      0          0       0         0      0       0           0
## 4          0      0          0       0         0      0       0           0
## 5          0      0          0       0         0      0       0           0
## 6          0      0          0       0         0      0       0           0
##   Venezuela Pakistan Burkina Faso Latvia Cuba Malta Qatar Samoa Ukraine
## 1         0        0            0      0    0     0     0     0       0
## 2         0        0            0      0    0     0     0     0       0
## 3         0        0            0      0    0     0     0     0       0
## 4         0        0            0      0    0     0     0     0       0
## 5         0        0            0      0    0     0     0     0       0
## 6         0        0            0      0    0     0     0     0       0
##   Colombia Cambodia Panama Georgia Dominican Republic Azerbaijan Armenia
## 1        0        0      0       0                  0          0       0
## 2        0        0      0       0                  0          0       0
## 3        0        0      0       0                  0          0       0
## 4        0        0      0       0                  0          0       0
## 5        0        0      0       0                  0          0       0
## 6        0        0      0       0                  0          0       0
##   Botswana Croatia Costa Rica Ghana Tunisia Rwanda Angola Monaco Puerto Rico
## 1        0       0          0     0       0      0      0      0           0
## 2        0       0          0     0       0      0      0      0           0
## 3        0       0          0     0       0      0      0      0           0
## 4        0       0          0     0       0      0      0      0           0
## 5        0       0          0     0       0      0      0      0           0
## 6        0       0          0     0       0      0      0      0           0
##    "Lao People Slovakia Gibraltar Liechtenstein Chad Iraq Serbia and Montenegro
## 1            0        0         0             0    0    0                     0
## 2            0        0         0             0    0    0                     0
## 3            0        0         0             0    0    0                     0
## 4            0        0         0             0    0    0                     0
## 5            0        0         0             0    0    0                     0
## 6            0        0         0             0    0    0                     0
##   Paraguay Animation Adventure Romance Comedy Action History Drama Crime
## 1        0         1         0       0      1      0       0     0     0
## 2        0         0         1       0      0      0       0     0     0
## 3        0         0         0       1      1      0       0     0     0
## 4        0         0         0       1      1      0       0     1     0
## 5        0         0         0       0      1      0       0     0     0
## 6        0         0         0       0      0      1       0     1     1
##   Fantasy Science Fiction Music Horror Family Mystery Thriller Western War
## 1       0               0     0      0      1       0        0       0   0
## 2       1               0     0      0      1       0        0       0   0
## 3       0               0     0      0      0       0        0       0   0
## 4       0               0     0      0      0       0        0       0   0
## 5       0               0     0      0      0       0        0       0   0
## 6       0               0     0      0      0       0        1       0   0
```

```
##   Documentary TV Movie Foreign English hit/not
## 1           0        0       0 English       1
## 2           0        0       0 English       1
## 3           0        0       0 English       1
## 4           0        0       0 English       1
## 5           0        0       0 English       1
## 6           0        0       0 English       1
```

Visualizations

```
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("readr")
library("stringr")
library("dplyr")
library("ggplot2")
```

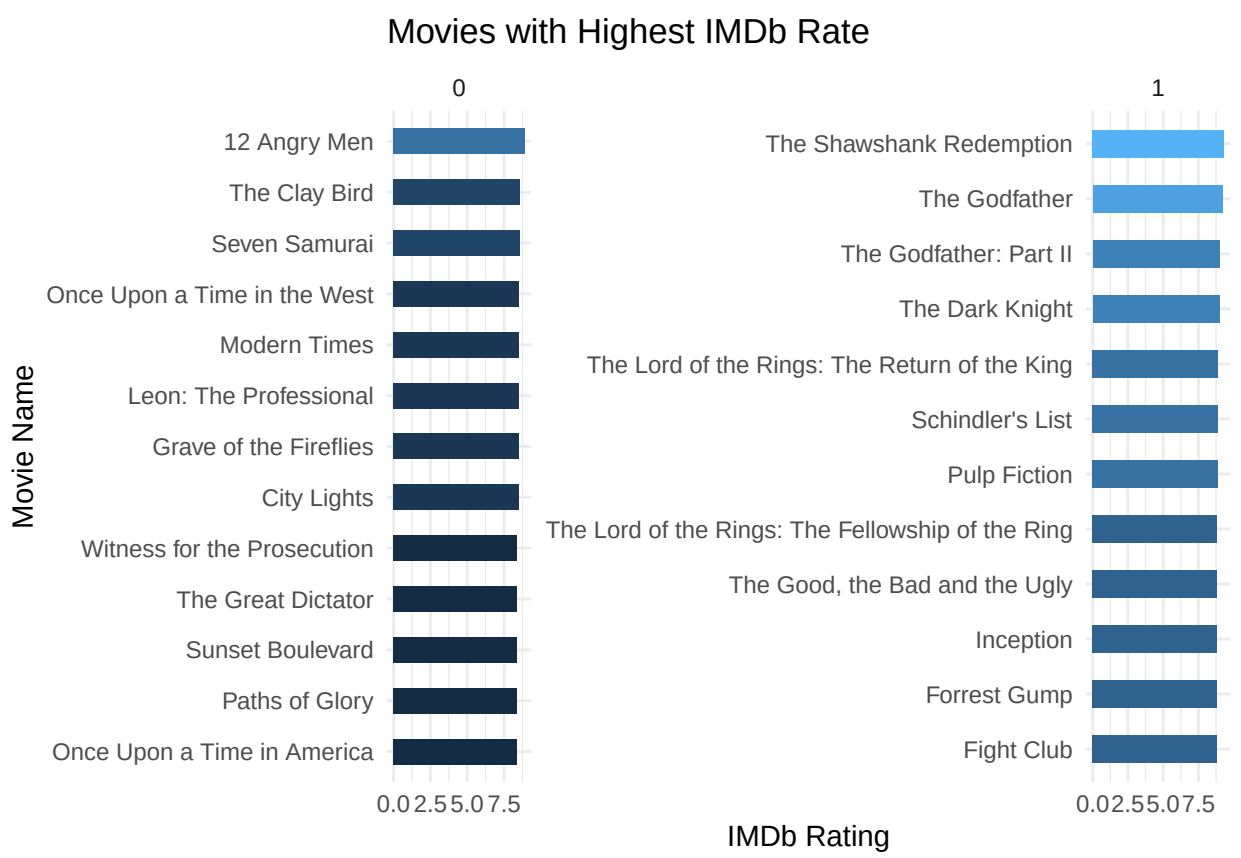
```
# Importing the tidy data file
setwd(getwd())
Cleaned_data <- as_tibble(read_csv("cleaned_merged.csv"))
```

```
## Rows: 8992 Columns: 147

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (7): genres, imdb_id, production_countries, original_language, title, ...
## dbl (139): popularity, runtime, vote_count, year, budget, worlwide_gross_inc...
## lgl   (1): adult

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# The top 20 Miovies with High IMDb Scores, For hit and not
# hit movies
top_imdb_movie <- Cleaned_data %>%
    group_by(`hit/not`) %>%
    top_n(10, wt = weighted_average_vote) %>%
    summarise(title, weighted_average_vote, `hit/not`) %>%
    arrange(desc(weighted_average_vote))
```

```
## `summarise()` has grouped output by 'hit/not'. You can override using the `.groups` argument.
```

```
top_imdb_movie %>%
    ggplot(aes(x = reorder(title, weighted_average_vote), y = weighted_average_vote,
        fill = weighted_average_vote)) + geom_col(width = 0.5,
    show.legend = FALSE) + facet_wrap(~`hit/not`, scales = "free") +
    coord_flip() + labs(x = "Movie Name", y = "IMDb Rating",
    title = "Movies with Highest IMDb Rate") + theme_minimal()
```

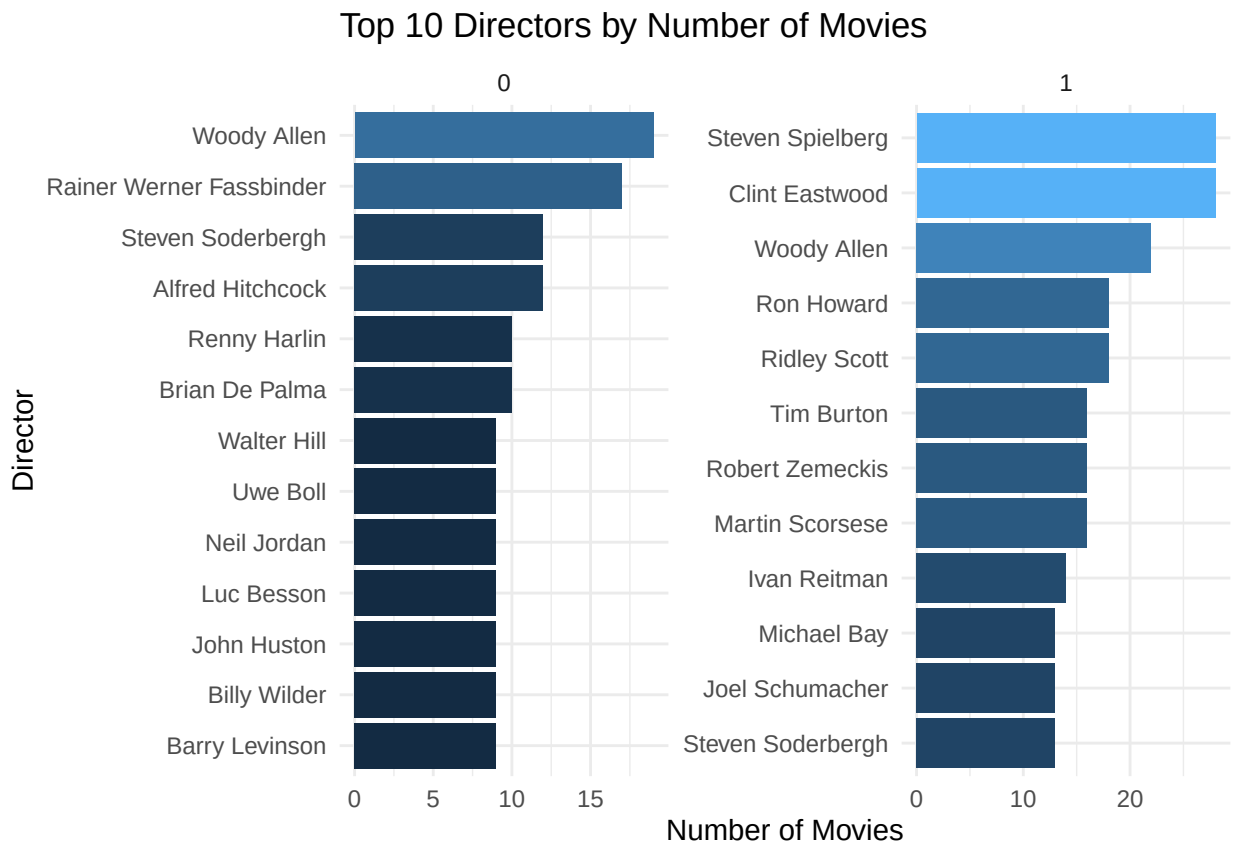



```
# From the above Figure we can see that the hit movies
# (hit/not=1), have demonstrated higher IMDb numbers in
# compared with the non-hit
```

```
# Top 10 directors for hit and not hit movies

top_directors <- Cleaned_data %>%
    group_by(`hit/not`) %>%
    count(director, sort = TRUE) %>%
    top_n(10) %>%
    ggplot(aes(x = reorder(director, n), y = n, fill = n)) +
    geom_col(show.legend = FALSE) + facet_wrap(~`hit/not`, scales = "free") +
    labs(x = "Director", y = "Number of Movies", title = "Top 10 Directors by Number of Movies") +
    coord_flip() + theme_minimal()
```

```
## Selecting by n
```

```
top_directors
```

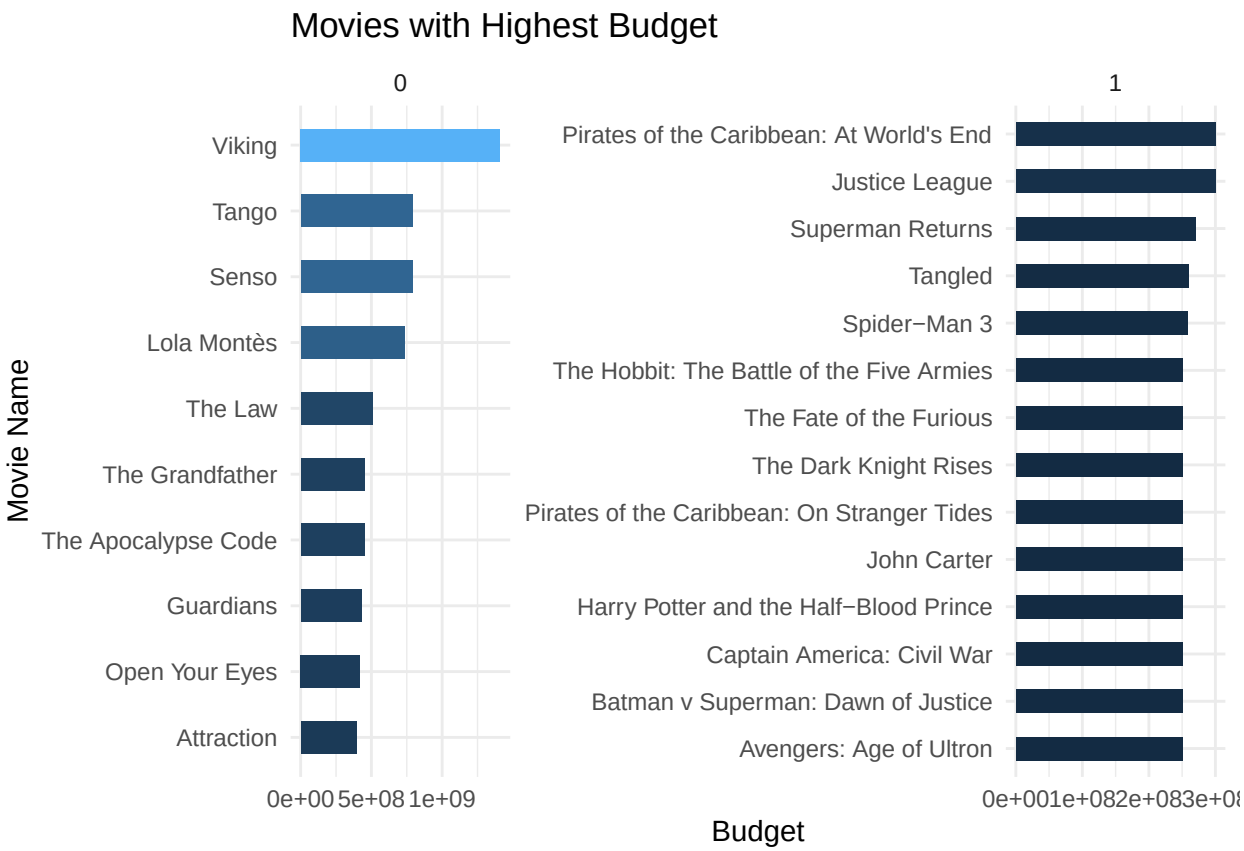



```
# Here we see that for hit moviies the number of movies the
# director has produced is more
```

```
# Movies with Highest budget faceted by hit or not hit

highest_budgets <- Cleaned_data %>%
    group_by(`hit/not`) %>%
    top_n(10, wt = budget) %>%
    summarise(title, budget, `hit/not`) %>%
    arrange(desc(budget))
```

```
## `summarise()` has grouped output by 'hit/not'. You can override using the `.groups` argument.
```

```
highest_budgets %>%
    ggplot(aes(x = reorder(title, budget), y = budget, fill = budget)) +
    geom_col(width = 0.5, show.legend = FALSE) + facet_wrap(~`hit/not`,
    scales = "free") + coord_flip() + labs(x = "Movie Name",
    y = "Budget", title = "Movies with Highest Budget") + theme_minimal()
```

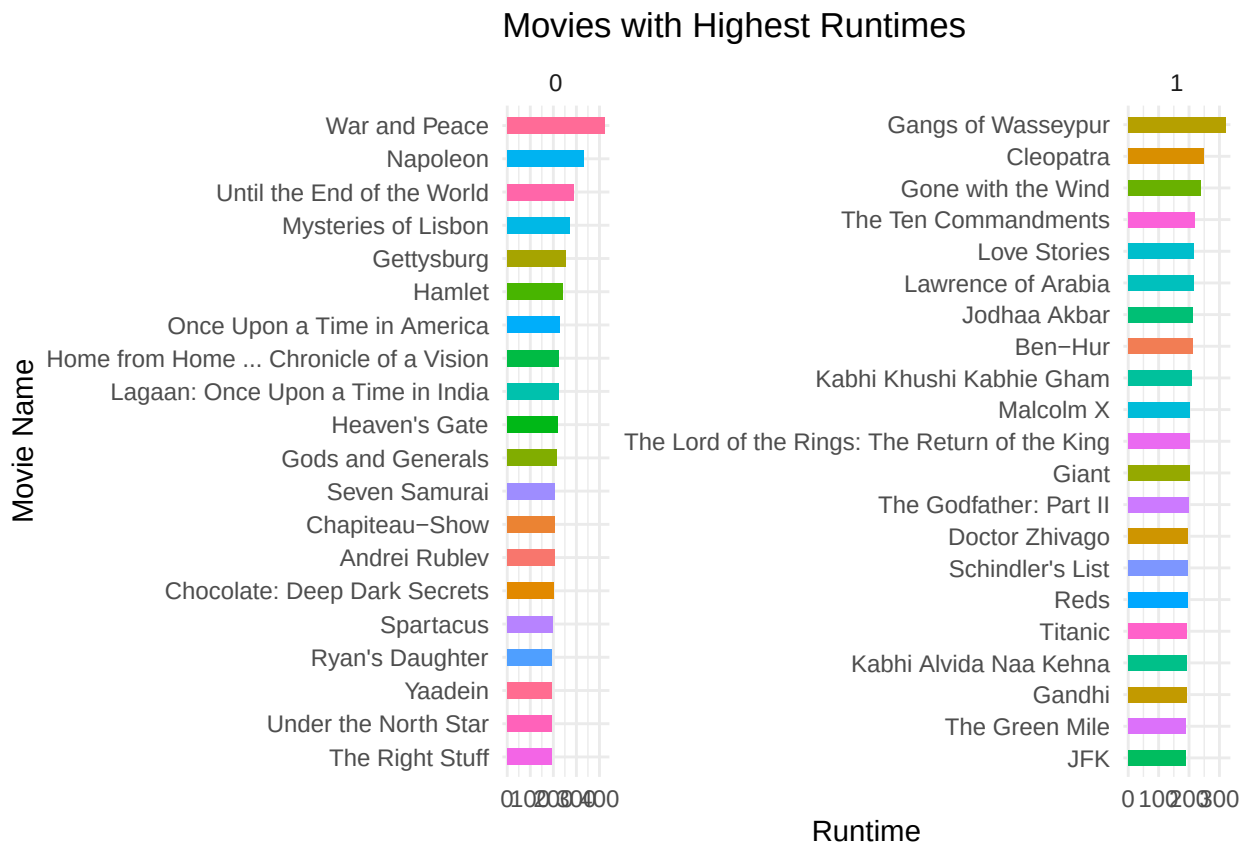



```
# For some reason we can see that non hit movies have
# higher movie budgets
```

```
# Movies with highest runtimes in terms of hit and non hit
highest_runtimes <- Cleaned_data %>%
    group_by(`hit/not`) %>%
    top_n(20, wt = runtime) %>%
    summarise(title, runtime, `hit/not`) %>%
    arrange(desc(runtime))
```

```
## `summarise()` has grouped output by 'hit/not'. You can override using the `.groups` argument.
```

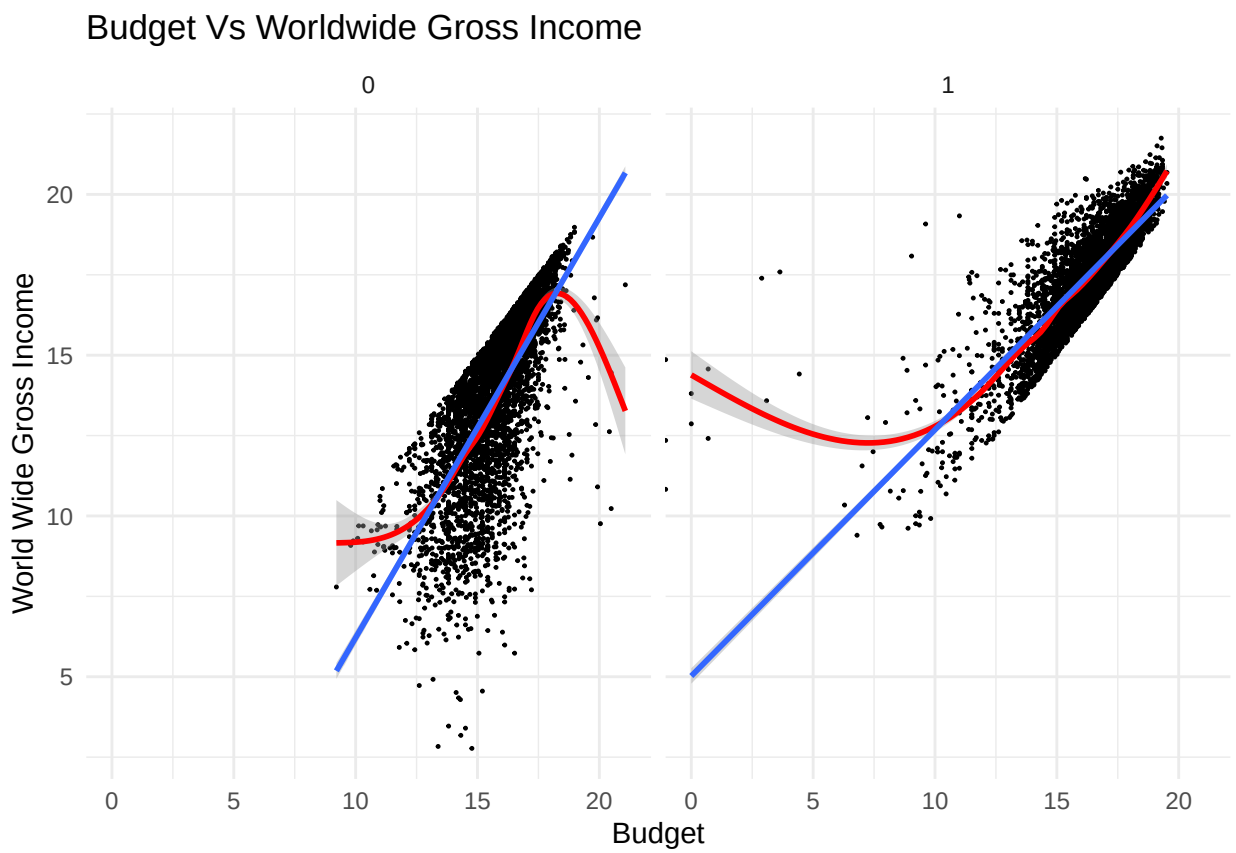
```
highest_runtimes %>%
    ggplot(aes(x = reorder(title, runtime), y = runtime, fill = title)) +
    geom_col(width = 0.5, show.legend = FALSE) + facet_wrap(~`hit/not`,
    scales = "free") + coord_flip() + labs(x = "Movie Name",
    y = "Runtime", title = "Movies with Highest Runtimes") +
    theme_minimal()
```




```
# We can see that the movie with the highest runtime was a
# nonhit movie.The rest of the movies weather it was a hit
# or not demonstrated an average runtime no more that 200.
```

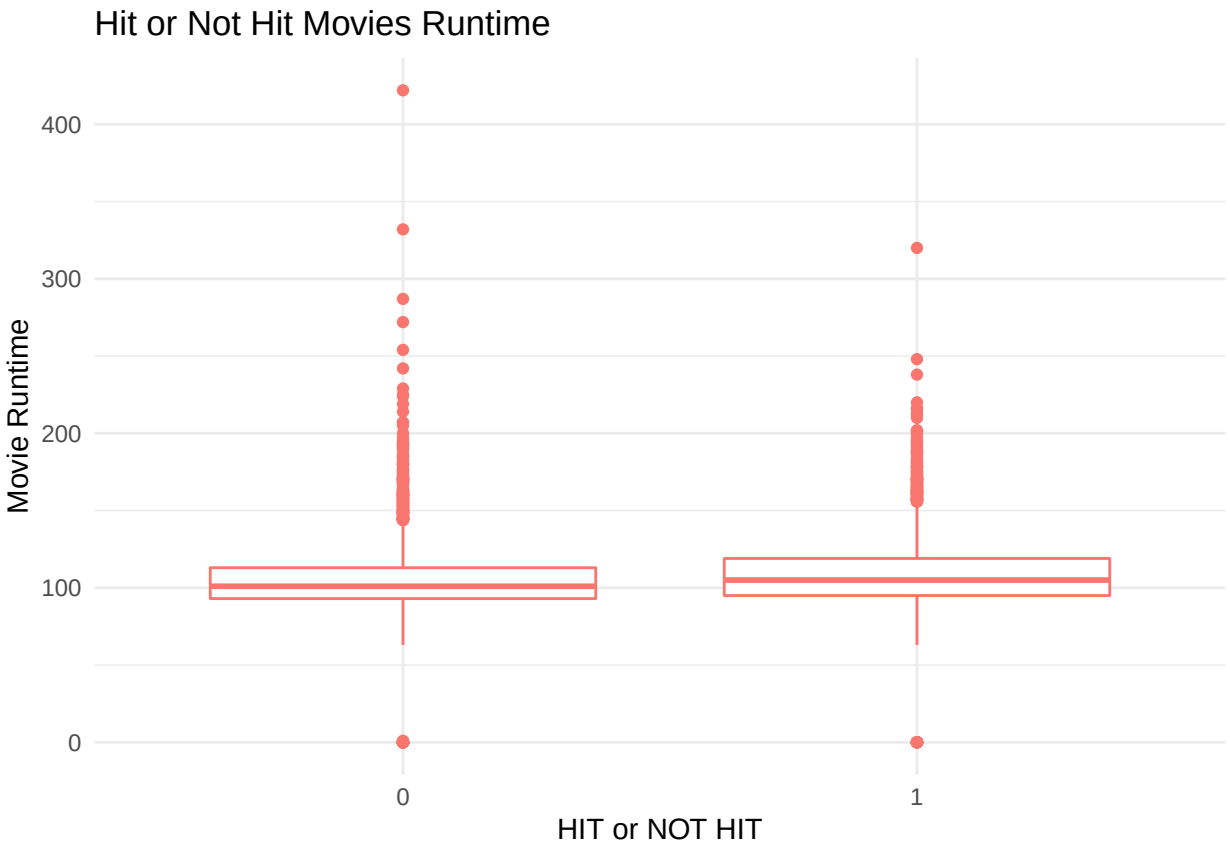
```
# Scatter plot to see if there is any relationship between
# budget and worldwide gross income faceted by hit or not
# hit movie
budget_grossincome <- ggplot(data = Cleaned_data, mapping = aes(x = log(budget),
    y = log(worlwide_gross_income))) + geom_point(size = 0.2) +
    labs(x = "Budget", y = "World Wide Gross Income", title = "Budget Vs Worldwide Gross Income") +
    facet_wrap(~`hit/not`) + geom_smooth(color = "red") + geom_smooth(method = lm) +
    theme_minimal()

budget_grossincome
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## `geom_smooth()` using formula 'y ~ x'
```




```
# From the above we can see that for both hit or not heat
# there is a positive relationship between budget and world
# wide gross income
```

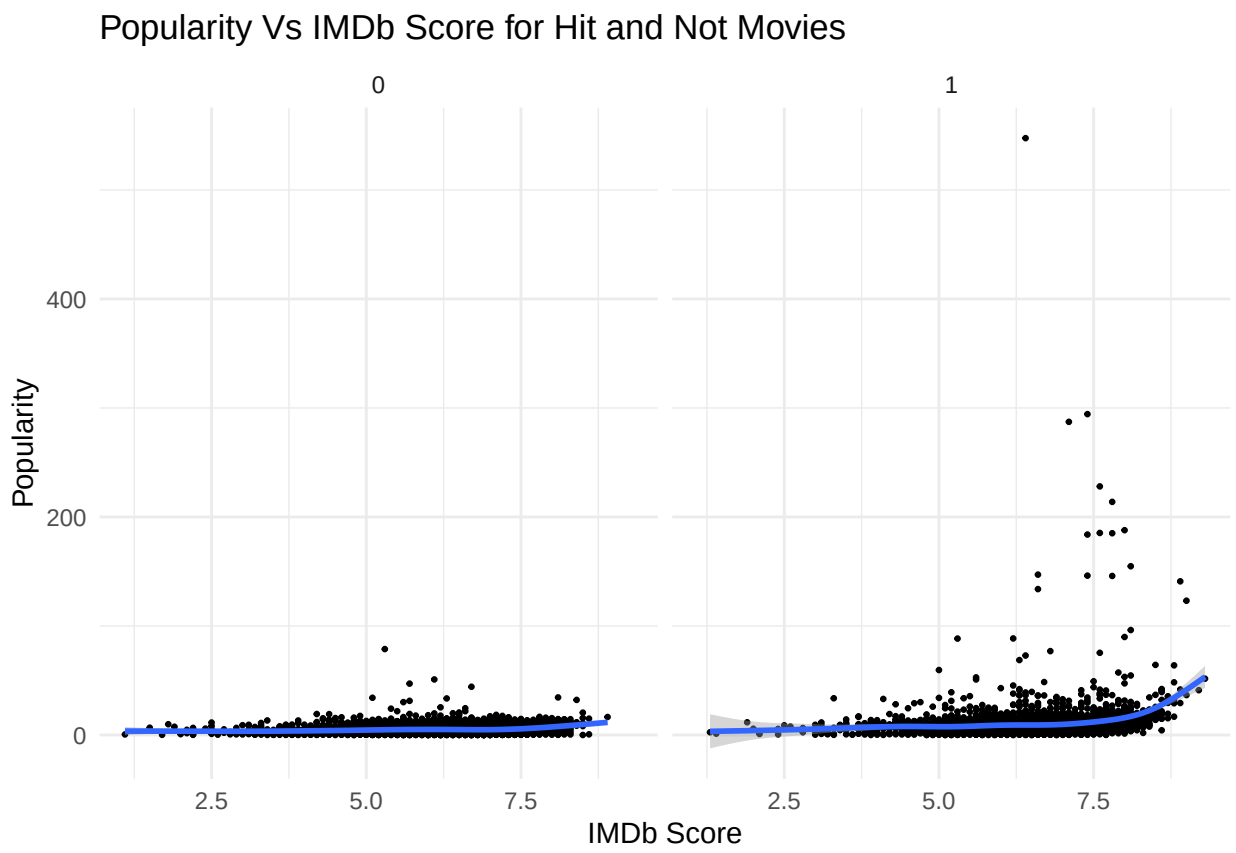
```
# runtime vs hit not hit boxplot
run_time_budget <- ggplot(data = Cleaned_data, mapping = aes(y = runtime,
    color = "`hit/not")) + geom_boxplot(aes(x = as.character(`hit/not`)),
    show.legend = "False") + labs(y = "Movie Runtime", x = "HIT or NOT HIT",
    title = "Hit or Not Hit Movies Runtime") + theme_minimal()
run_time_budget
```




```
# From the boxplot we can see that that most nonhit movies
# have demonstrated an average run time about 100 and for
# hit movies the average run time was a bit higher, both
# demonstrated right skewed model.
```

```
# IMDb Score and popularity #weird
Vote_average_popularity <- ggplot(data = Cleaned_data, mapping = aes(x = weighted_average_vote,
    y = popularity)) + geom_point(size = 0.5) + facet_wrap(~`hit/not`) +
    labs(x = "IMDb Score", y = "Popularity", title = "Popularity Vs IMDb Score for Hit and Not Movies")
    geom_smooth() + theme_minimal()
Vote_average_popularity
```

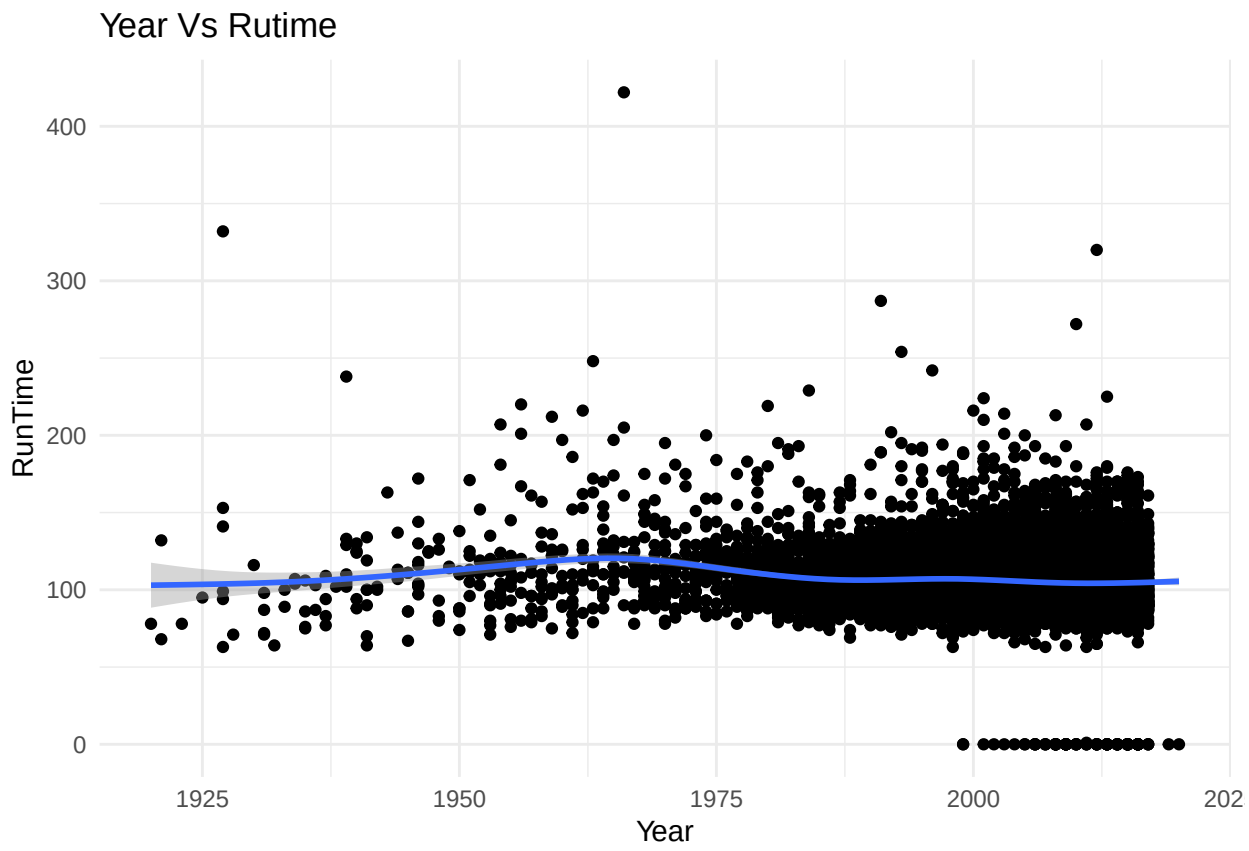
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
# For both hit and non nit movies it showed a weird
# correlation between population and IMDb Score, where the
# popularity was relatively low, but it demonstrated a
# higher value for hit movies with higher
```

```
# runtimevs released year
runtime_vs_release_year <- ggplot(data = Cleaned_data, mapping = aes(x = year,
    y = runtime)) + geom_point() + labs(x = "Year", y = "RunTime",
    title = "Year Vs Rutime") + geom_smooth() + theme_minimal()

runtime_vs_release_year
```

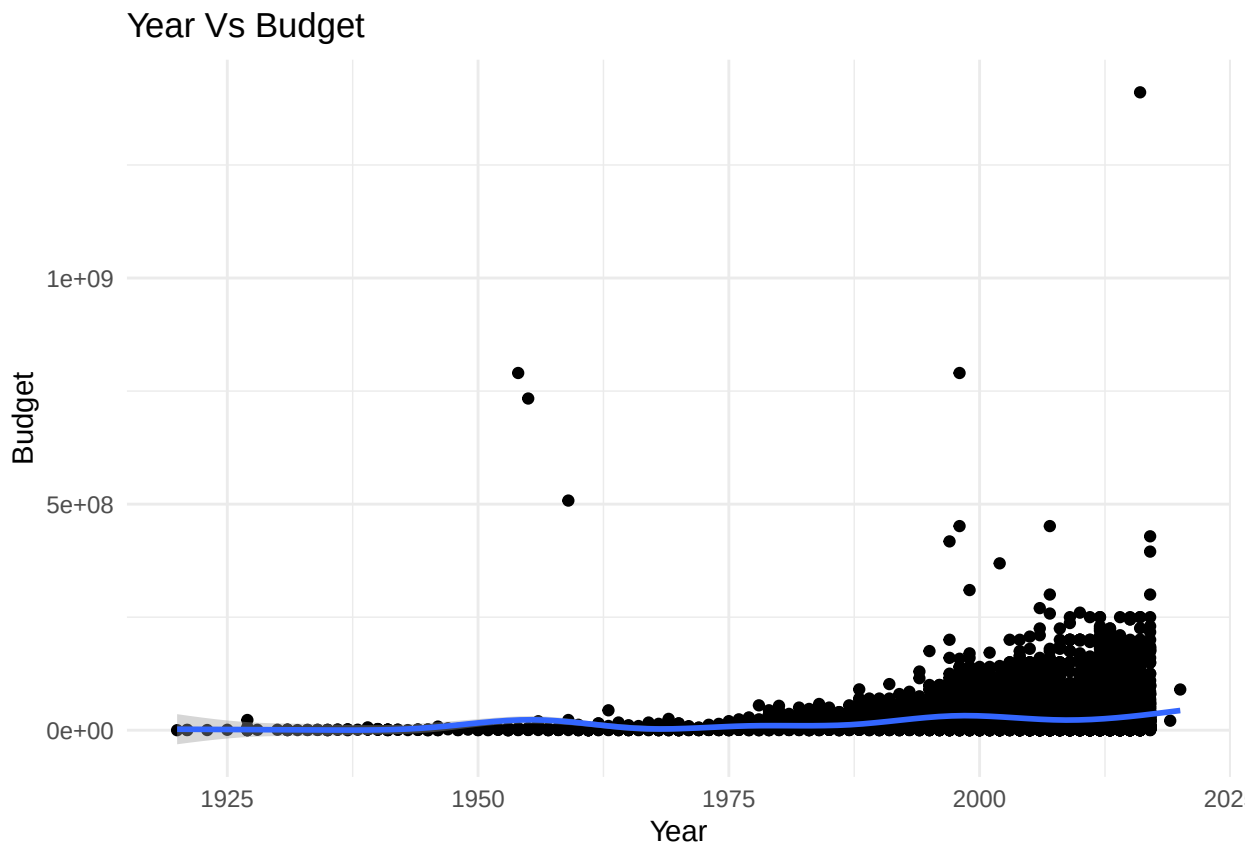
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
# We can see as the years increase we see higher runtimes
```

```
# budgetvs release year
budget_vs_release_year <- ggplot(data = Cleaned_data, mapping = aes(x = year,
    y = budget)) + geom_point() + labs(x = "Year", y = "Budget",
    title = "Year Vs Budget") + geom_smooth() + theme_minimal()

budget_vs_release_year
```

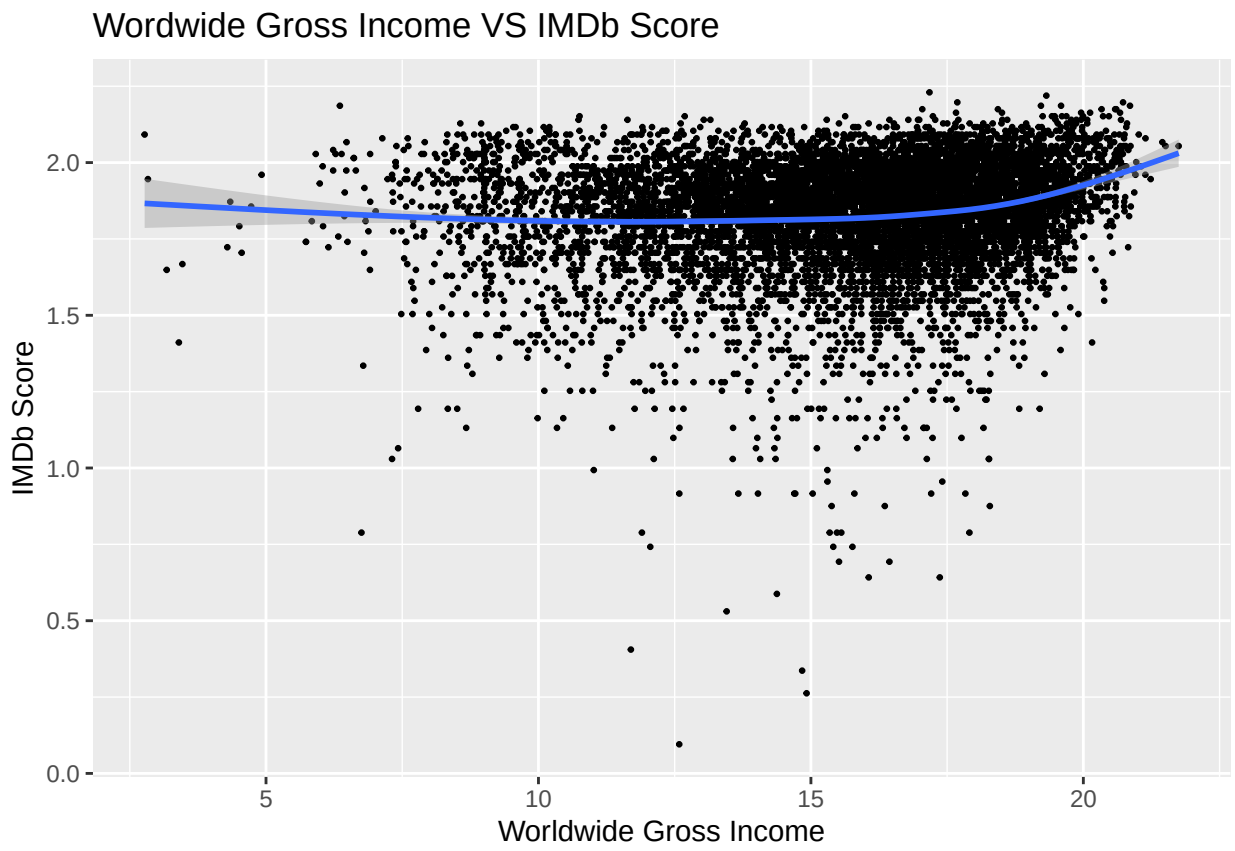
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



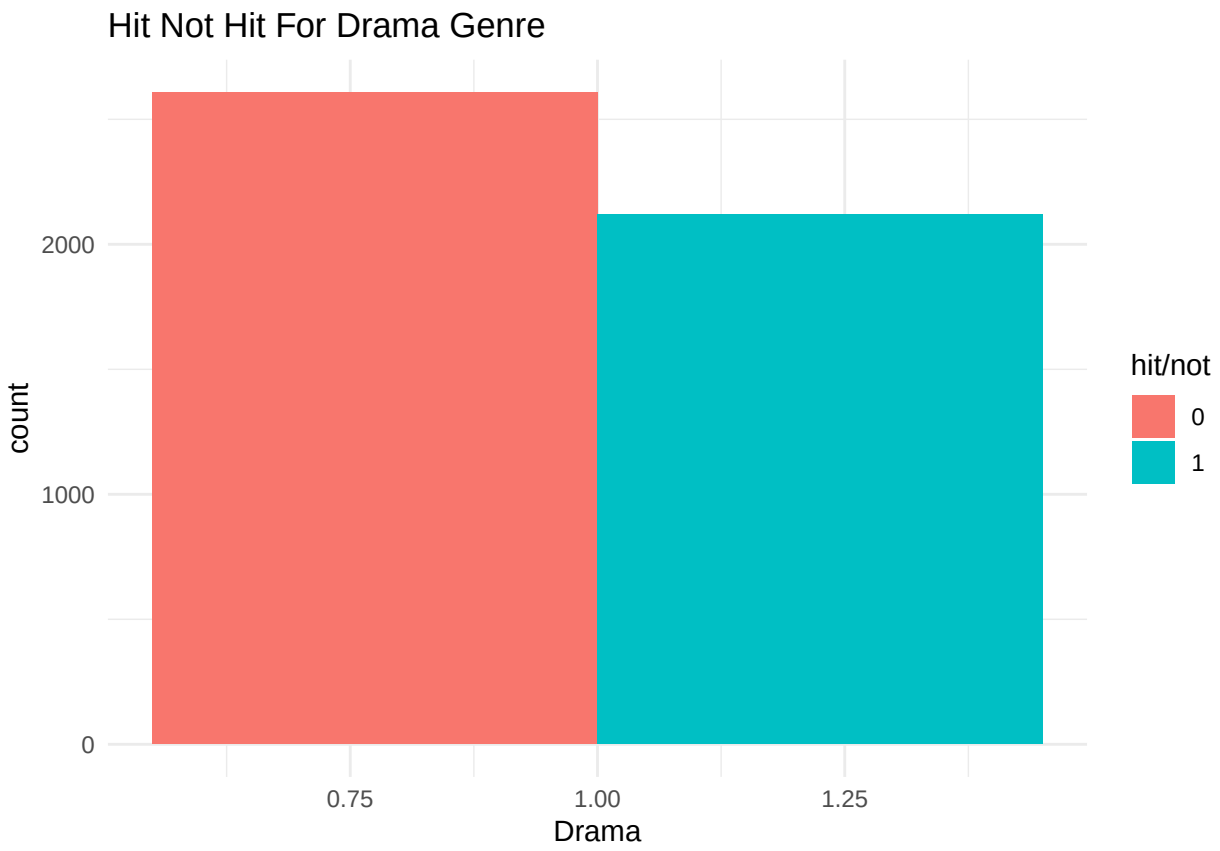

```
# we also see that as the years increase the budget
# increases
```

```
# IMDB Score vs world_wide income
world_vote <- ggplot(data = Cleaned_data, mapping = aes(x = log(worlwide_gross_income),
    y = log(weighted_average_vote))) + geom_point(size = 0.5) +
    labs(x = "Worldwide Gross Income", y = "IMDb Score", title = "Wordwide Gross Income VS IMDb Score")
    geom_smooth()

world_vote
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

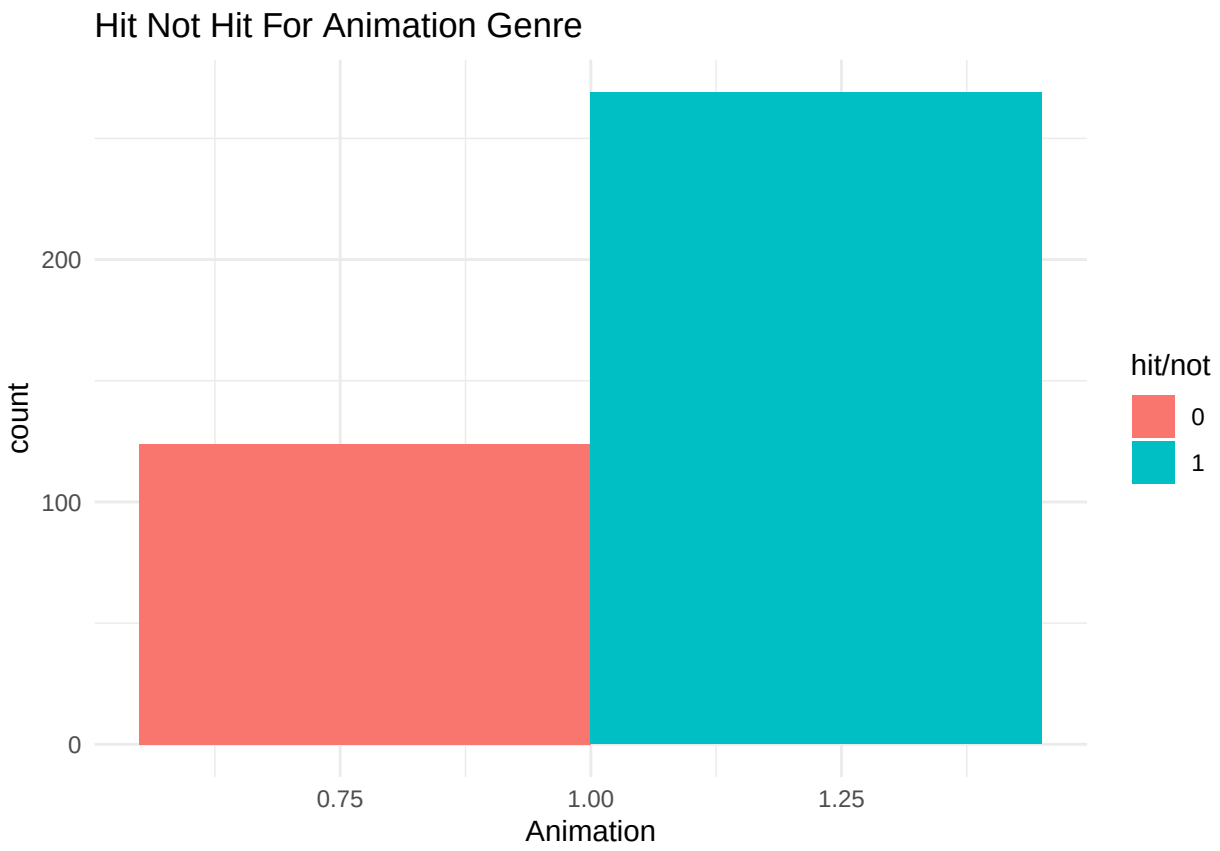



```
# hit not hit agisnt drama
Drama_hit_not <- Cleaned_data %>%
    filter(Drama == 1) %>%
    ggplot(aes(x = Drama, fill = as.factor(`hit/not`))) + geom_bar(position = "dodge") +
    scale_fill_discrete(name = "hit/not") + labs(title = "Hit Not Hit For Drama Genre") +
    theme_minimal()

Drama_hit_not
```

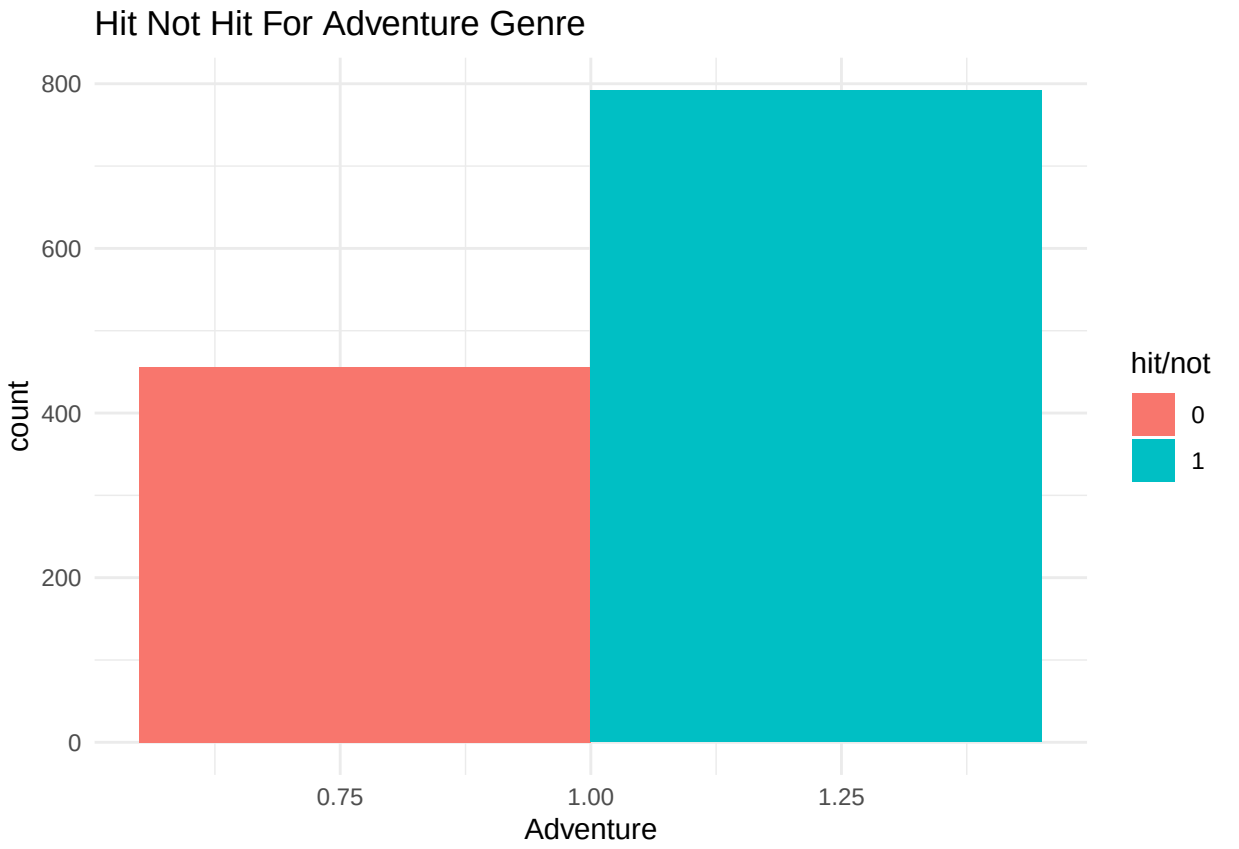



```
# we can see for Drama (genre) that movies that were not
# hit were more in the drama genre
```

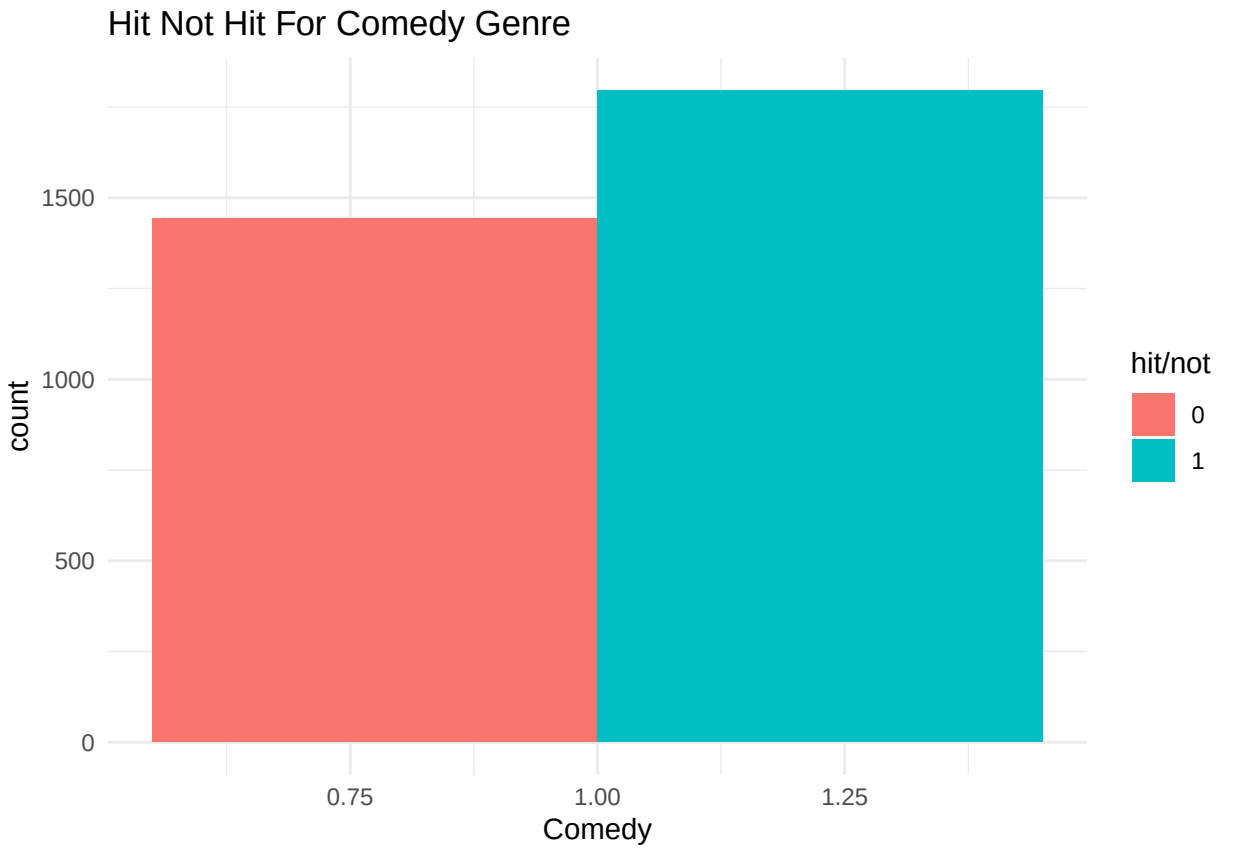
```
Animation_hit_not <- Cleaned_data %>%
    filter(Animation == 1) %>%
    ggplot(aes(x = Animation, fill = as.factor(`hit/not`))) +
    geom_bar(position = "dodge") + scale_fill_discrete(name = "hit/not") +
    labs(title = "Hit Not Hit For Animation Genre") + theme_minimal()

Animation_hit_not
```



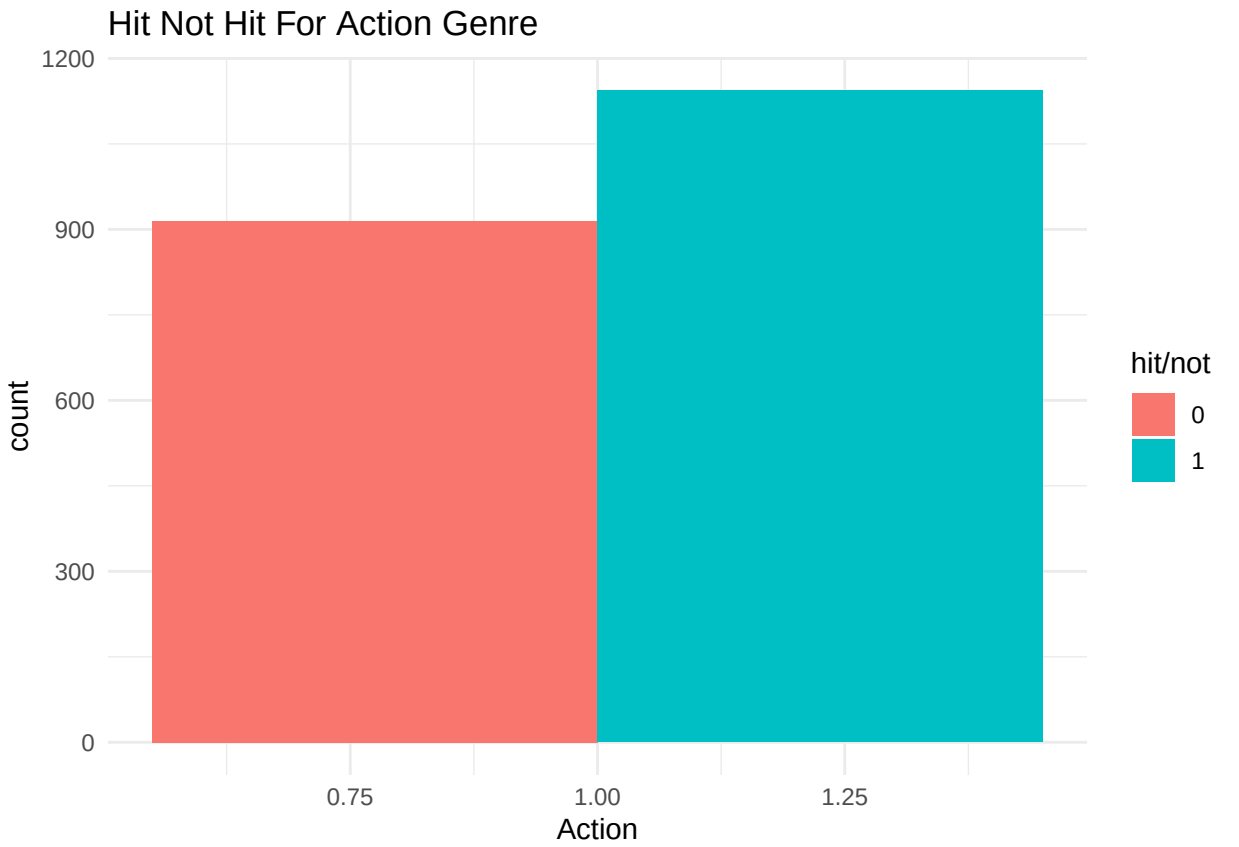

```
# hot not hit for Adventure
Adventure_hit_not <- Cleaned_data %>%
    filter(Adventure == 1) %>%
    ggplot(aes(x = Adventure, fill = as.factor(`hit/not`))) +
    geom_bar(position = "dodge") + scale_fill_discrete(name = "hit/not") +
    labs(title = "Hit Not Hit For Adventure Genre") + theme_minimal()

Adventure_hit_not
```



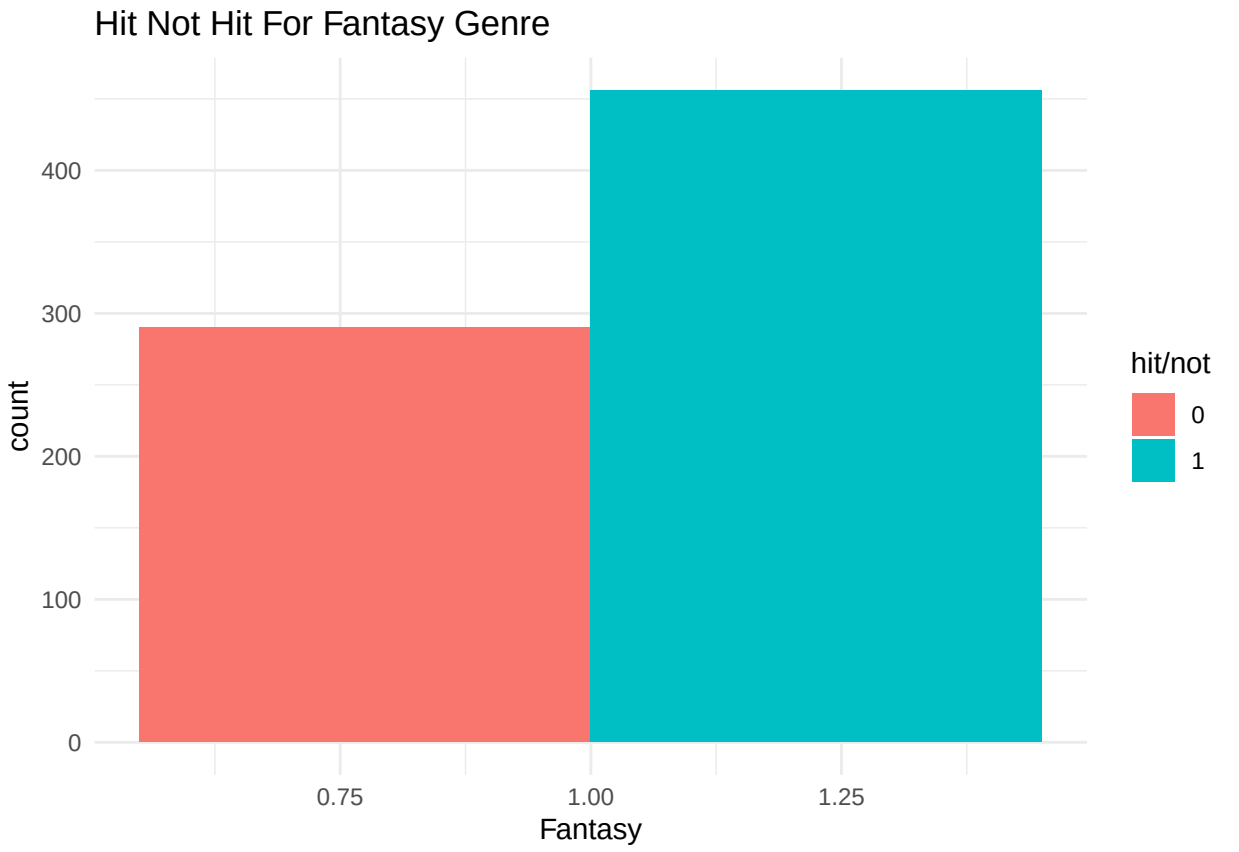

```
Comedy_hit_not <- Cleaned_data %>%
    filter(Comedy == 1) %>%
    ggplot(aes(x = Comedy, fill = as.factor(`hit/not`))) + geom_bar(position = "dodge") +
    scale_fill_discrete(name = "hit/not") + labs(title = "Hit Not Hit For Comedy Genre") +
    theme_minimal()
Comedy_hit_not
```



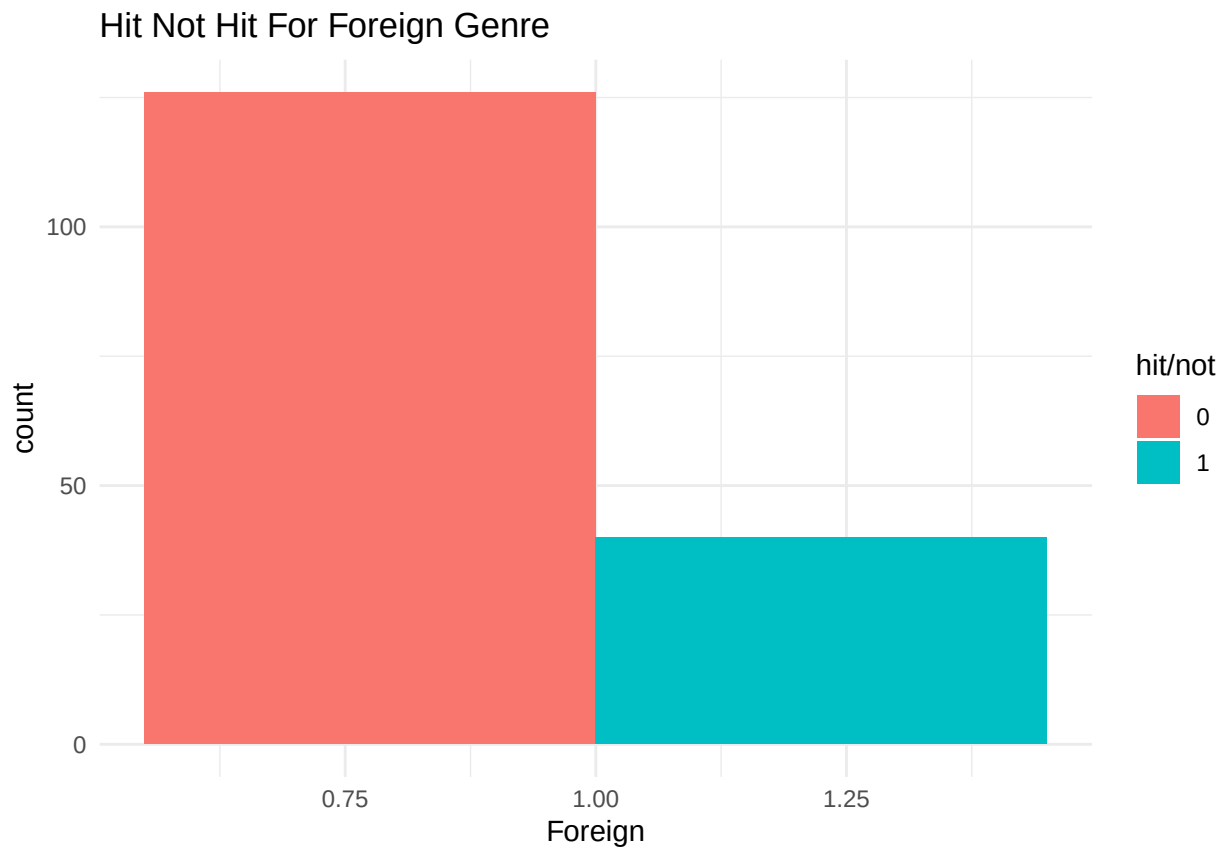

```
Action_hit_not <- Cleaned_data %>%
    filter(Action == 1) %>%
    ggplot(aes(x = Action, fill = as.factor(`hit/not`))) + geom_bar(position = "dodge") +
    scale_fill_discrete(name = "hit/not") + labs(title = "Hit Not Hit For Action Genre") +
    theme_minimal()

Action_hit_not
```



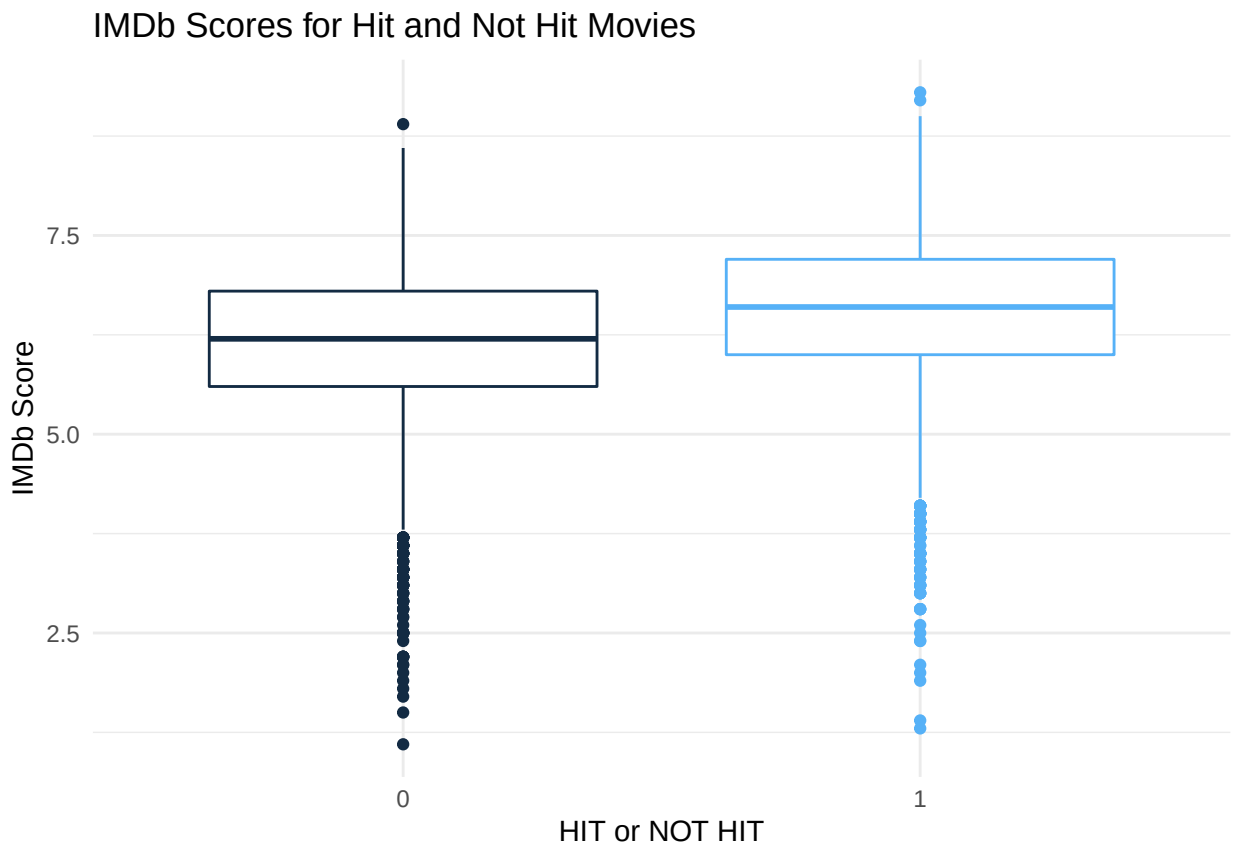

```
Fantasy_hit_not <- Cleaned_data %>%
    filter(Fantasy == 1) %>%
    ggplot(aes(x = Fantasy, fill = as.factor(`hit/not`))) + geom_bar(position = "dodge") +
    scale_fill_discrete(name = "hit/not") + labs(title = "Hit Not Hit For Fantasy Genre") +
    theme_minimal()

Fantasy_hit_not
```




```
Foreign_hit_not <- Cleaned_data %>%
    filter(Foreign == 1) %>%
    ggplot(aes(x = Foreign, fill = as.factor(`hit/not`))) + geom_bar(position = "dodge") +
    scale_fill_discrete(name = "hit/not") + labs(title = "Hit Not Hit For Foreign Genre") +
    theme_minimal()

Foreign_hit_not
```

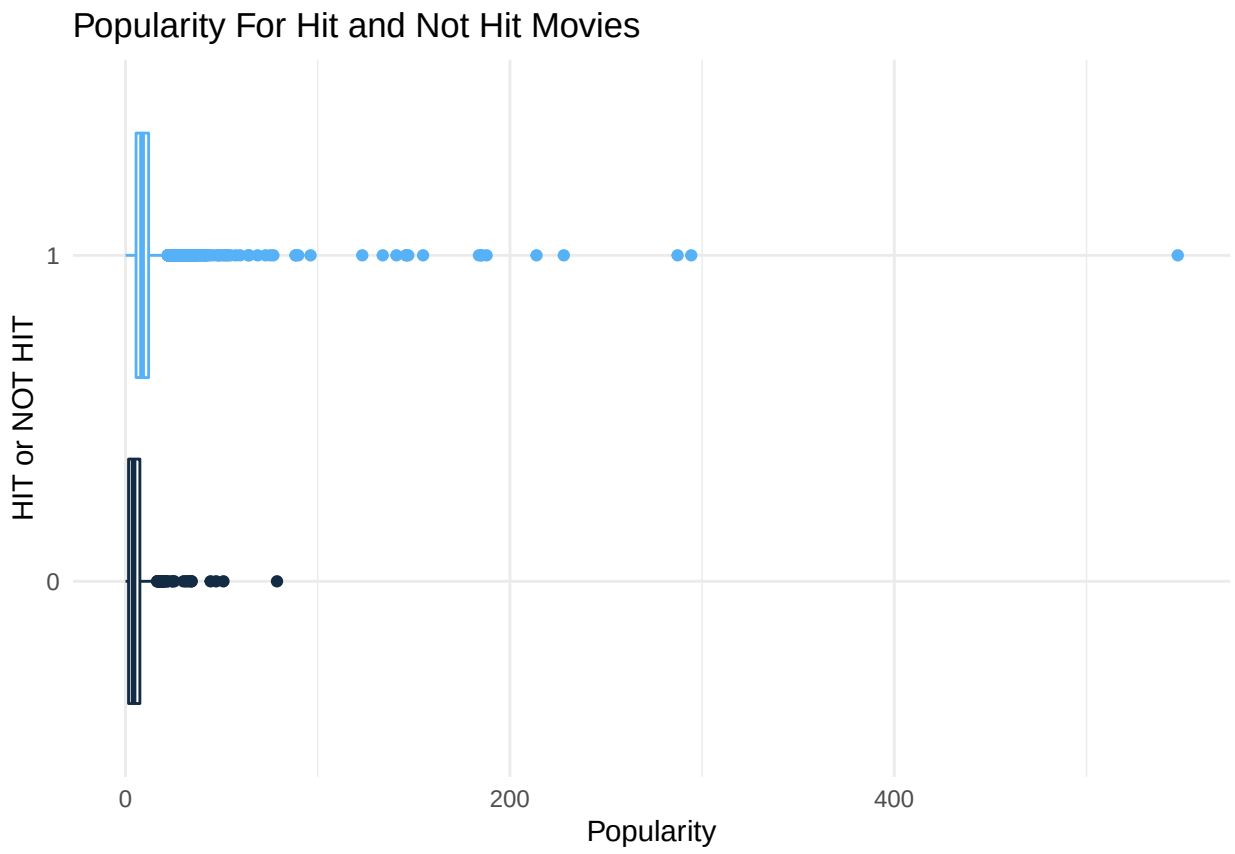



```
Cleaned_data <- Cleaned_data %>%
    mutate(ratio = ifelse(budget == 0, 0, as.numeric(worlwide_gross_income)/as.numeric(budget)))


# hit not hit with IMDb Score
vote_hit_not <- ggplot(data = Cleaned_data, mapping = aes(y = weighted_average_vote)) +
    geom_boxplot(aes(x = as.character(`hit/not`), color = `hit/not`),
        show.legend = FALSE) + labs(y = "IMDb Score", x = "HIT or NOT HIT",
    title = "IMDb Scores for Hit and Not Hit Movies") + theme_minimal()
vote_hit_not
```

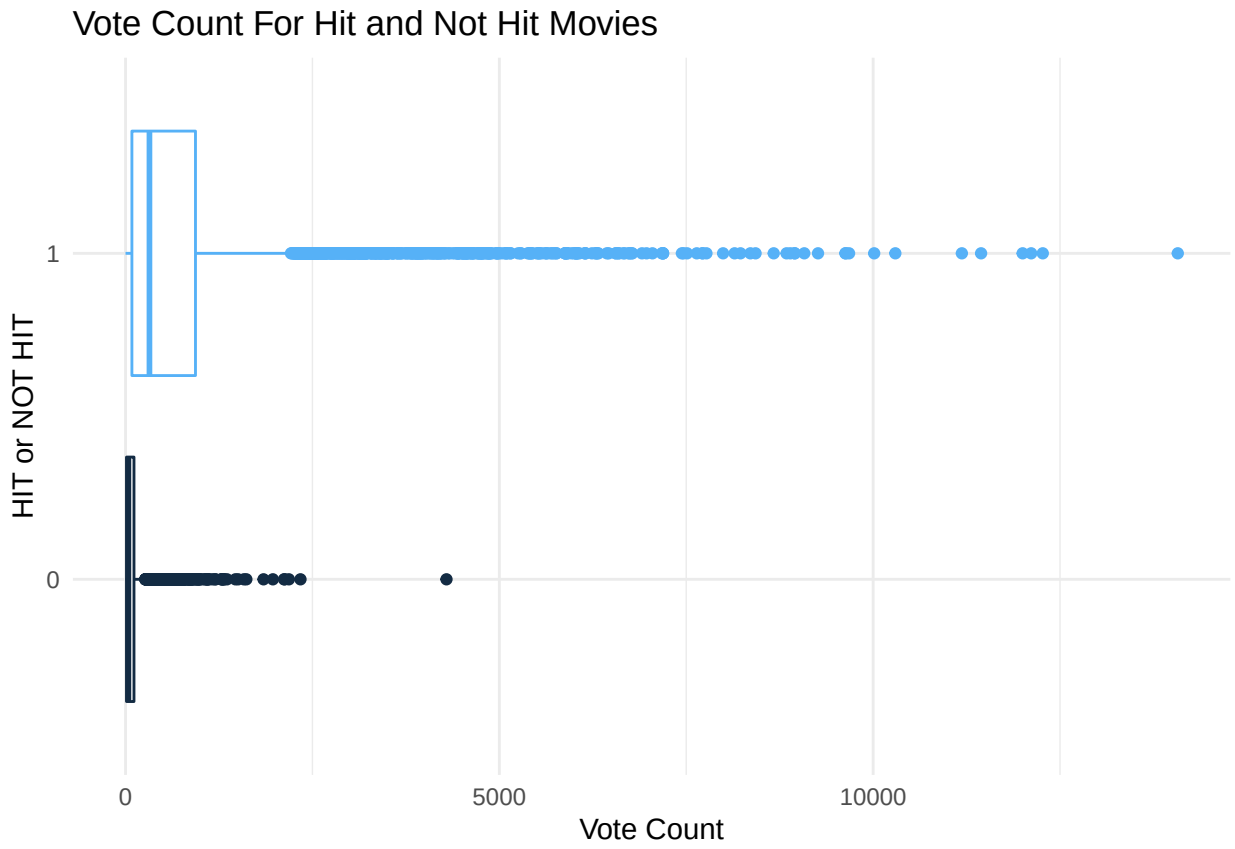



```
# we can see that hit movies have higher imdb scores
```

```
# hit not hit with popularity
popularity_hit_not <- ggplot(data = Cleaned_data, mapping = aes(x = popularity)) +
    geom_boxplot(aes(y = as.character(`hit/not`), color = `hit/not`),
        show.legend = FALSE) + labs(x = "Popularity", y = "HIT or NOT HIT",
    title = "Popularity For Hit and Not Hit Movies") + theme_minimal()

popularity_hit_not
```




```
Votecount_hit_not <- ggplot(data = Cleaned_data, mapping = aes(x = vote_count)) +
    geom_boxplot(aes(y = as.character(`hit/not`), color = `hit/not`),
        show.legend = FALSE) + labs(x = "Vote Count", y = "HIT or NOT HIT",
    title = "Vote Count For Hit and Not Hit Movies") + theme_minimal()


Votecount_hit_not
```


Vote Count For Hit and Not Hit Movies
HIT or NOT HIT
1
0
0
5000
10000
Vote Count

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

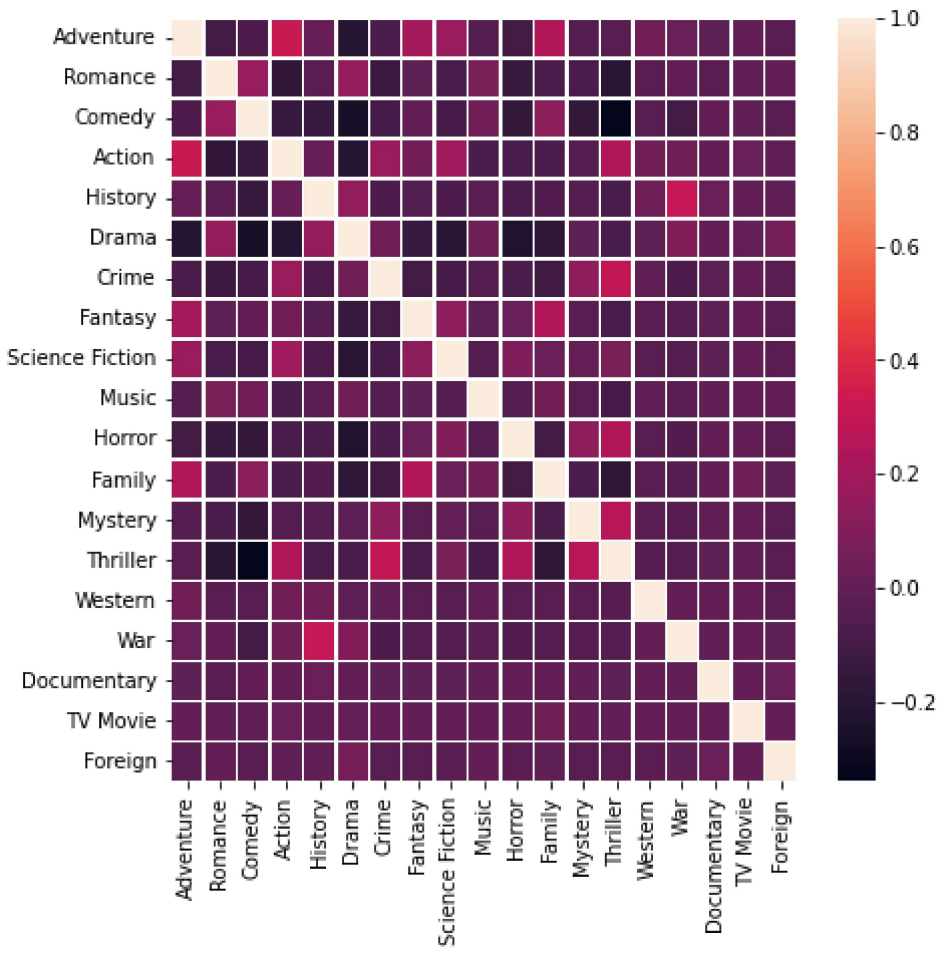
```
# Pandas is used to convert our csv to workable dataframe.
import pandas as pd

# Seabord is an advanced visualising tool, used here , for Heat maps.
import seaborn as sns

# Matplotlib is used to plot data into graphs.
import matplotlib.pyplot as plt

# Using Sklearn to model and retrieve evaluation metrics.
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import precision_score, f1_score, roc_curve
from sklearn.metrics import auc, recall_score, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
```

```
# Importing the cleaned Dataset into a pandas variable called df
df  = pd.read_csv("/content/drive/MyDrive/IDMP PROJECT/Final  Dataset/Final Dataset.csv")
```

```
# Accessing only the genres related columns from the dataframe
genres = df[df.columns[126:146]]
```

```
# Using the matplotlib library to retireve a subplot of size 7x7
fig, ax = plt.subplots(figsize=(7,7))
# Using the Seaborn library to plot the heatmap on our subplot
sns.heatmap(genres.corr(), linewidths=.5, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7632591ad0>
```




```
# Converting the Bool column "adult" into 1s and 0s
df["adult"].replace(True,'1', inplace=True)
df['adult'].replace(False,'0', inplace=True)
```

```
  # Retriving the correlation values of only the target variable
  # corr() gives the correlation values of all variables against each other.
  # Since df is a pandas dataframe, we use column name(hit/not)
  # to retrieve the correlation vlaues relative to that column

  cor = df.corr()
  cor_tar = cor["hit/not"]

  #Using a threshold value of 0.04 we choose the columns only that
  # have correlation greater than threshold

```

```
threshold = 0.04
rel_cor = cor_tar[abs(cor_tar) > threshold ]
```

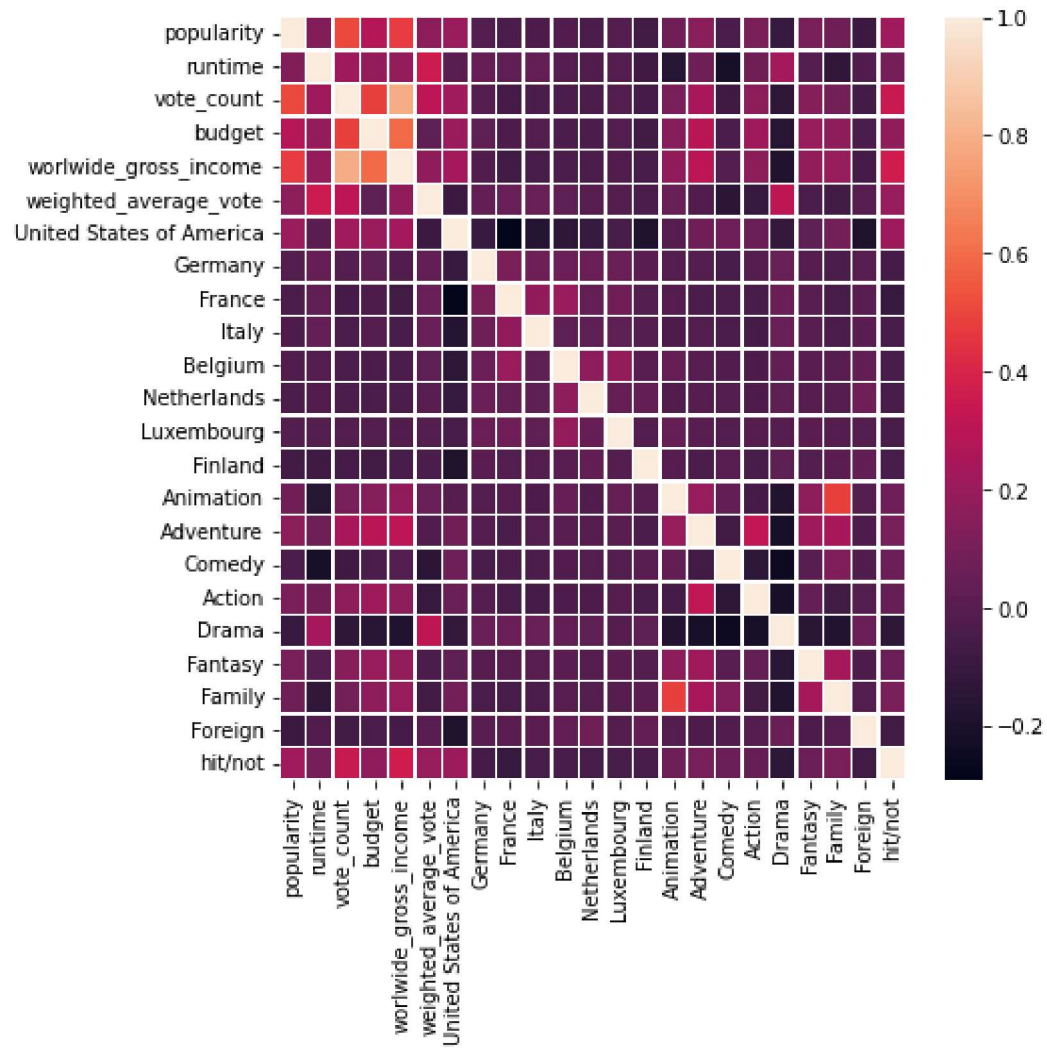
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8992 entries, 0 to 8991
Columns: 147 entries, adult to hit/not
dtypes: float64(4), int64(135), object(8)
memory usage: 10.1+ MB
```

```
# rel_cols only contain those columns that have greater correlation with our
# target variable.

rel_cols = ["popularity","runtime","vote_count","budget","worlwide_gross_income",
            "weighted_average_vote","United States of America","Germany","France",
            "Italy","Belgium","Netherlands","Luxembourg","Finland","Animation",
            "Adventure","Comedy","Action","Drama","Fantasy","Family","Foreign",
            "hit/not"]
# Filtering out the columns that are irrelevant to our model
non_rel = []
for col in df.columns:
    if col not in rel_cols:
        non_rel.append(col)

# using drop() to drop those columns from our dataframe
df = df.drop(non_rel, axis=1)
```

```
# creating a Heat map for relevant Columns
fig, ax = plt.subplots(figsize=(7,7))
sns.heatmap(df.corr(), linewidths=.5, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f763c010710>
```




```
## Splitting data into Test and Training set

X = df.iloc[:, :-1:]
y = df.iloc[:, -1]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=0)
```

```
# Evaluation Metrics
# Using Sklearns to calculate metrics
# We are calculating accuracy, precision, F1-score, AUC and Specificity
def eval_metrics(y_test, y_pred):
  accuracy = accuracy_score(y_test, y_pred)
  cm = confusion_matrix(y_test, y_pred)
  precision = precision_score(y_test, y_pred)
  recall = recall_score(y_test, y_pred)
  f1 = f1_score(y_test, y_pred)
  fpr, tpr, thresholds = roc_curve(y_test, y_pred, pos_label=1)
```

```
11   A = auc(fpr, tpr)
12   roc = roc_auc_score(y_test, y_pred)
13   tn, fp, fn, tp = cm.ravel()
14   specificity = tn / (tn+fp)
15   return accuracy*100, precision, recall, f1, A, specificity
16
```

Logistic Regression

```
 1 # Using Logistic Regression model from Sklearn to model our data.
 2 # Since it is a basic model, the we used mse to calculate error.
 3
 4 # loading the LogisticRegression module in to lr
 5 lr = LogisticRegression()
 6
 7 # Fitting the model to our train set
 8 lr.fit(x_train, y_train)
 9
10 # Using the trained model to predict out X_Test
11 pred = lr.predict(x_test)
12
13 # Calculating our evaluation metrics using sklearn build-in package.
14 acc_lr,precision_lr, rec_lr, f1_lr, areaUnderCurve_lr, spec_lr= eval_metrics(y_test, pred)
15
16 # Printing out the Evaluation Metrics
17 print(f"The metrics for Logistic regression are: \n Accuracy: {acc_lr}% ",
18       f"\nPrecision: {precision_lr} \nRecall: {rec_lr} \nF1 score: {f1_lr} ",
19       f"\nArea Under Curve: {areaUnderCurve_lr}  \nSpecificity: {spec_lr}")
20
```

```
The metrics for Logistic regression are:
 Accuracy: 100.0%
Precision: 1.0
Recall: 1.0
F1 score: 1.0
Area Under Curve: 1.0
Specificity: 1.0
```

KNN:

```
 1 # Using KneighborsClassifier model from Sklearn to model our data.
 2 # Since it is a basic model, the we used mse to calculate error.
 3
 4 # loading the KNeighborsClassifier module in to knn
 5 knn = KNeighborsClassifier()
 6
 7 # Fitting the model to our train set
 8 knn.fit(x_train, y_train)
 9
10 #  Using the trained model to predict out X_Test
11 y_pred_knn = knn.predict(x_test)
12
13 # Calculating our evaluation metrics using sklearn build-in package.
14 acc_knn, precision_knn, rec_knn, f1_knn, areaUnderCurve_knn, spec_knn = eval_metrics(y_test, y_pred_knn)
15
16 # Printing out the Evaluation Metrics
17 print(f"The metrics for KNN are: \n Accuracy: {acc_knn}%  \nPrecision: {precision_knn}"+
18       f" \nRecall: {rec_knn} \nF1 score: {f1_knn} \nArea Under Curve: {areaUnderCurve_knn}"+
19       f" \nSpecificity: {spec_knn}")
20
```

```
The metrics for KNN are:
 Accuracy: 99.27737632017788%
Precision: 0.9913606911447084
Recall: 0.9945828819068255
F1 score: 0.9929691725256895
Area Under Curve: 0.9927252309077507
Specificity: 0.9908675799086758
```

Decision Tree:

```
 1   # Using KneighborsClassifier model from Sklearn to model our data.
 2   # For Decision Tree we take Cross-Entropy Loss as loss function.
 3   # We set that minimum split samples to be 5
 4   # minimum number of leaf nodes is 6
 5   # and we set the max_features to auto to facilitate the model to random skip a few features.
 6   # Random State is set as 50.
 7
 8   # loading the KNeighborsClassifier module in to dt with hyperparameters
 9   dt = tree.DecisionTreeClassifier(criterion='entropy', min_samples_split=5,
10                                    min_samples_leaf=6, max_features='auto',
```

```
                                  random_state=50)

# Fitting the model to our train set
dt.fit(x_train, y_train)

#  Using the trained model to predict out X_Test
y_pred_dt = dt.predict(x_test)

# Calculating our evaluation metrics using sklearn build-in package.
acc_dt, precision_dt, rec_dt, f1_dt, areaUnderCurve_dt, spec_dt = eval_metrics(y_test, y_pred_dt)

# Printing out the Evaluation Metrics
print(f"The metrics for Decision Tree are: \n Accuracy: {acc_dt}%",
      f" \nPrecision: {precision_dt} \nRecall: {rec_dt} \nF1 score:",
      f" {f1_dt} \nArea Under Curve: {areaUnderCurve_dt}",
      f" \nSpecificity: {spec_dt}")

```

```
The metrics for Decision Tree are:
 Accuracy: 85.38076709282934%
Precision: 0.8395061728395061
Recall: 0.8840736728060672
F1 score:  0.8612137203166227
Area Under Curve: 0.8529957405126226
Specificity: 0.821917808219178
```

Random Forest

```
# Using RandomForestClassifier model from Sklearn to model our data.
# For Random Forest, we take Cross-Entropy Loss as loss function.
# We set that minimum split samples to be 5
# minimum number of leaf nodes is 6
# and we set the max_features to auto to facilitate the model to random skip a few features.
# Random State is set as 50.

# loading the KNeighborsClassifier module in to rf with hyperparameters
rf = RandomForestClassifier(n_estimators=50,
                            criterion='entropy', min_samples_split=5,
                            min_samples_leaf=6, max_features='auto',
                            random_state=50)

# Fitting the model to our train set
rf.fit(x_train, y_train)

#  Using the trained model to predict out X_Test
y_pred_rf = rf.predict(x_test)

# Calculating our evaluation metrics using sklearn build-in package.
acc_rf, precision_rf, rec_rf, f1_rf, areaUnderCurve_rf, spec_rf = eval_metrics(y_test, y_pred_rf)

# Printing out the Evaluation Metrics
print(f"The metrics for Random Forest are: \n Accuracy: {acc_rf}%"+
      f" \nPrecision: {precision_rf} \nRecall: {rec_rf} \nF1 score: {f1_rf} "+
      f"\nArea Under Curve: {areaUnderCurve_rf} \nSpecificity: {areaUnderCurve_rf}")

```

```
The metrics for Random Forest are:
 Accuracy: 95.27515286270149%
Precision: 0.9554347826086956
Recall: 0.952329360780065
F1 score: 0.9538795442213781
Area Under Curve: 0.9527628539060143
Specificity: 0.9527628539060143
```