Exploring +1000 Movie Ratings

Roel Peters

25 september 2016

Abstract

Movies as a cultural product offer a readable glance into the zeitgeist of … who am I kidding. This is not a scientific paper, so I will not bore anyone with expensive words from the scientific community. Au contraire, this is my own first document written using R, Rstudio, RMarkdown & Knitr. It offers an overview of the first 1000 movies I have watched in my life. I look at my favorite genres, actors and directors. Finally, I build a machine learning model that should be able to predict with a 80% accuracy if I will like a movie or not. Enjoy.

Introduction

During my life I always kept track of the movies I watched. When I was a teenager, I built a MS Access database. Somewhere around 2011 - however, I could be mistaken - I decided to turn to IMDb's rating system. I no longer had to manually add genres & actors and it came in handily with an app for my smartphone.

This year, I finally reached rating number 1000. It has been my plan for a long time that this rating was reserved 'The Godfather'. Well, things didn't go according to plan. and movie 1000 became Fruitvale Station which portraits the last hours of Oscar Grant III's life, before he became a symbol of police brutality in the United States.

In this document I will explore the movies I watched by answering several questions regarding several features such as genre, country and actors. I also propose a prediction model that can help me guide my future movie selections. Maybe living inside Eli Pariser's Filter Bubble is fun if I build a prediction model myself?

Happy reading!

Exploring the dataset

Ratings export

On this page one can find an overview of my ratings. When I am logged in under my username I can do a CSV export. This file has been saved under ratings.csv and can be find in the repository.

```
loadRatings <- function(file = "ratings.csv") {
    read.csv("ratings.csv",header=T,sep=",")
}
dataFromRoel <- loadRatings()
```

OMDb API

Using the OMDb API I was able to enrich the data a little bit with the Motion Picture Association of America film rating, the writer, a list of actors, language, country and the awards.

```
loadOMDB <- function(df = loadRatings()) {
    dfOMDB <- data.frame()
    if (!file.exists("omdbdata.csv")) {
        for (i in 1:nrow(df)) {
            id <- df[i,2]
            url <- gsub(" ","%20",paste("http://www.omdbapi.com/?i=",as.character(id), sep = ""))
            message(paste("Requesting data for",id <- df[i,6],"- Item ",i,"of",nrow(df),collapse=" "))
            omdbData <- fromJSON(url)
            if (!is.null(omdbData$totalSeasons)) { omdbData$totalSeasons <- NULL }
            if (i == 1) {
                dfOMDB <- as.data.frame(omdbData,stringsAsFactors=F)
            } else {
                dfOMDB[i,] <- as.character(omdbData)
            }
        }
        write.csv(dfOMDB,"omdbdata.csv",row.names=F)
    }
    dfOMDB <- read.csv("omdbdata.csv",header=TRUE,stringsAsFactors = F)
}
dataFromOMDB <- loadOMDB()
```

In the following section we merge the data from the IMDb export and from the OMDb API.

```
getData <- function(data1 = dataFromRoel,data2 = dataFromOMDB) {
    data1 <- data1[c("const","created","You.rated")]
    dataset <- merge(data1,data2,by.x="const",by.y="imdbID")
}
ds <- getData()
```

Feature building

Several features such as the directors, the actors, the countries and the genres are lists with comma's. The following script transforms these lists into dummy variables - a 0 or 1 when a specific actor plays in the movie, when a director has worked on the film, when the movie originates from a specific country and when a movie is attributed to a specific genre.

```
makeLong <- function(dataset = df,col="Genre") {
    for (i in 1:nrow(dataset)) {
        types <- strsplit(as.character(dataset[i,col]),", ")[[1]]
        for (j in 1:length(types)) {
            dataset[i,paste(col,types[j],sep="_")] <- 1
        }
    }
    colnames(dataset) <- gsub("[^[:alnum:]///' ]", "", colnames(dataset))
    dataset[is.na(dataset)] <- 0
    dataset
}
ds <- makeLong(ds)
ds <- makeLong(ds,"Director")
ds <- makeLong(ds,"Actors")
ds <- makeLong(ds,"Country")
```

Next, we're doing some cleaning of the data:

- Replace character-type "N/A"'s with real NA's
- Replacing spaces & dashes in column names
- Renaming some columns that were transformed to dummy variable columns to avoid mistakes
- Removing series. We're only interest in movies.
- Removing unnecessary columns we do not need as features.
- Remove dummy variables where the frequency of '1' is less than 6
- Adding the column we want to predict in a later phase: is my rating more than 6/10 or less?

```r
# Remove fake NA's
ds[ds == 'N/A'] <- NA

# Make important covariates & outcomes numeric
ds$imdbVotes <- gsub(",","",ds$imdbVotes)
ds[,c("Yourated","Metascore","imdbVotes","imdbRating","Year")] <- ds[,c("Yourated","Metascore","imdb

# Replace dashes & spaces in column names
colnames(ds) <- gsub("-","",colnames(ds))
colnames(ds) <- gsub(" ","",colnames(ds))

# Rename columns where values became dummy variables
names(ds)[names(ds) == 'Director'] <- 'MultipleD'
names(ds)[names(ds) == 'Actors'] <- 'MultipleA'
names(ds)[names(ds) == 'Genre'] <- 'MultipleG'
names(ds)[names(ds) == 'Country'] <- 'MultipleC'

# We're not interested in series, only in movies
moviesOnly <- function(df) {
    df <- df[df$Type=="movie",]
}
ds <- moviesOnly(ds)

# Remove unnecessary columns
selectColumns <- function(df) {
    df <- subset(df,select = -c(const,created,Rated,Released,Runtime,MultipleG,MultipleD,Writer,Mul
}
ds <- selectColumns(ds)

# We're only interested in dummy variables that are 1 more than 5 times.
addTreshold <- function(df) {
    sl <- c()
    for (i in 1:ncol(df)) {
        if (grepl("^(Actors|Director|Genre|Country)",names(df[i]))) {
            if (sum(df[,i])<5) {
                sl[i] <- FALSE
            } else {
                sl[i] <- TRUE
            }
        } else {
            sl[i] <- TRUE
        }
    }
    df <- df[,sl]
}
ds <- addTreshold(ds)

# Change NA's to zeroes in dummy variables
removeNA <- function(df) {
    suffix <- c("^Genre","^Actors","^Director","^Country")
    for (j in 1:length(suffix)) {
        genreCols <- grepl(suffix[j],names(df))
        for (i in 1:sum(genreCols)) {
            column <- df[,genreCols][i]
            column[is.na(column)] <- 0
            df[,genreCols][i] <- column
        }
    }
    df
}
ds <- removeNA(ds)

ds$sixHigh <- ifelse(ds$Yourated > 6,"HIGH","LOW")
ds <- ds %>% mutate(diffRating = Yourated - imdbRating)

# There is one movie with an error which generated NA's. Let's remove that from the analysis.
ds <- ds[complete.cases(ds),]
```

Exploratory Analysis

In this section I will try to answer several questions regarding the features of my dataset:

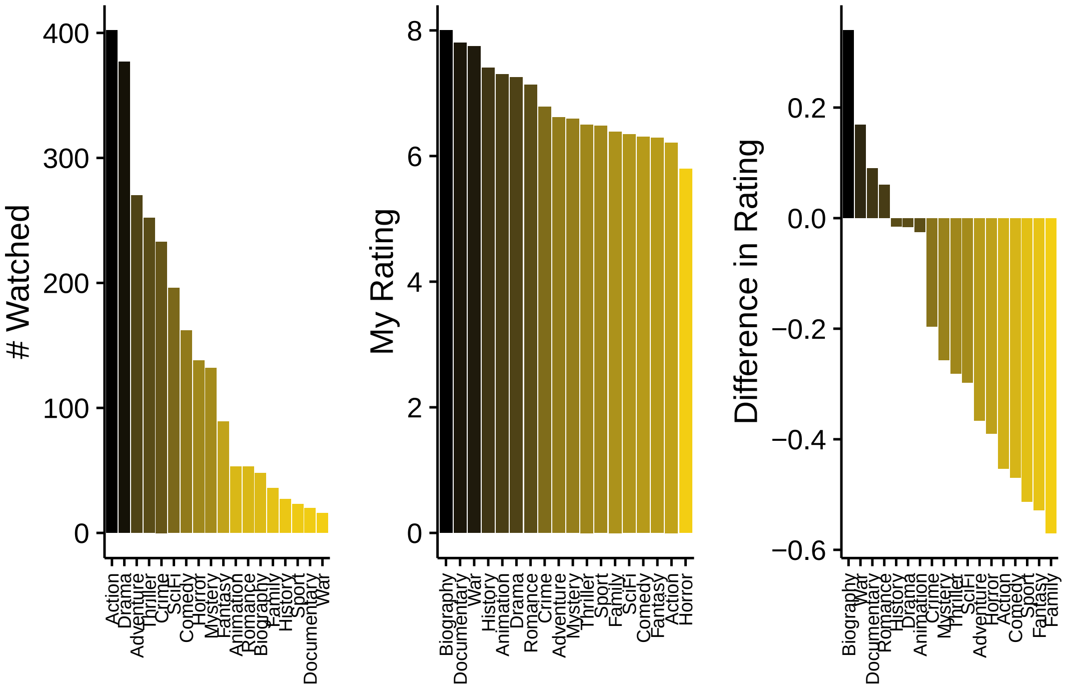
- According to my ratings, which is my favourite movie genre?
- According to my ratings, who are my favourite actors?
- According to my ratings, who are my favourite directors?

There's three ways to answer these questions. In the first interpretation we simply look at the frequency of each genre, actor and director.In the second interpretation we account for my given rating, but we ignore that there is a bandwagon effect, so we do not account for the IMDb average rating. In the third and last interpretation we acknowledge the existence of influence from the outside world and we look at the difference of my ratings with those of the other people on IMDb.

We do this by splitting each and every bar chart into three panels. The first panel displays the frequency, the second my rating and the third panel gives the difference in rating with the IMDb ratings.

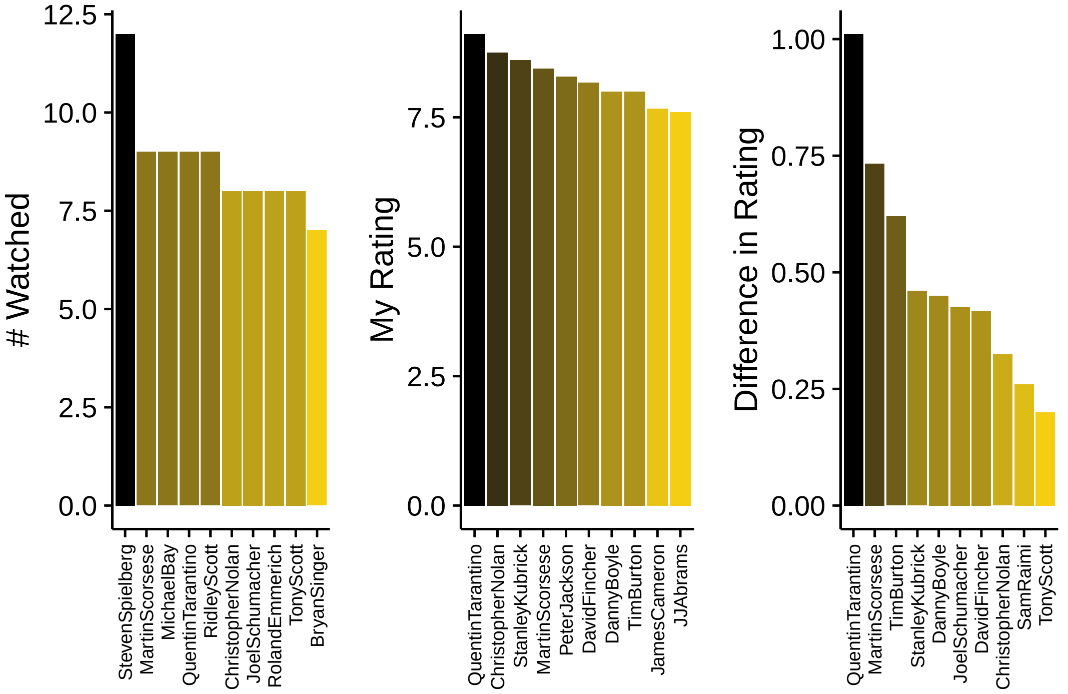
Favorite Genres

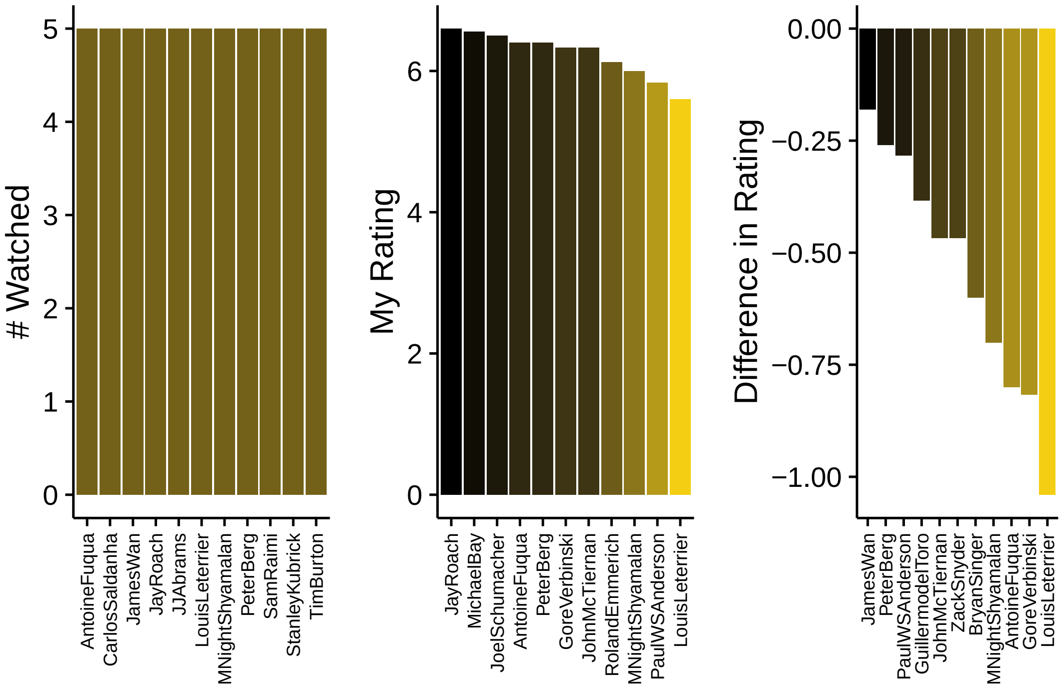
The following figure is a bar chart that ranks each genre by average rating.



Favorite Directors

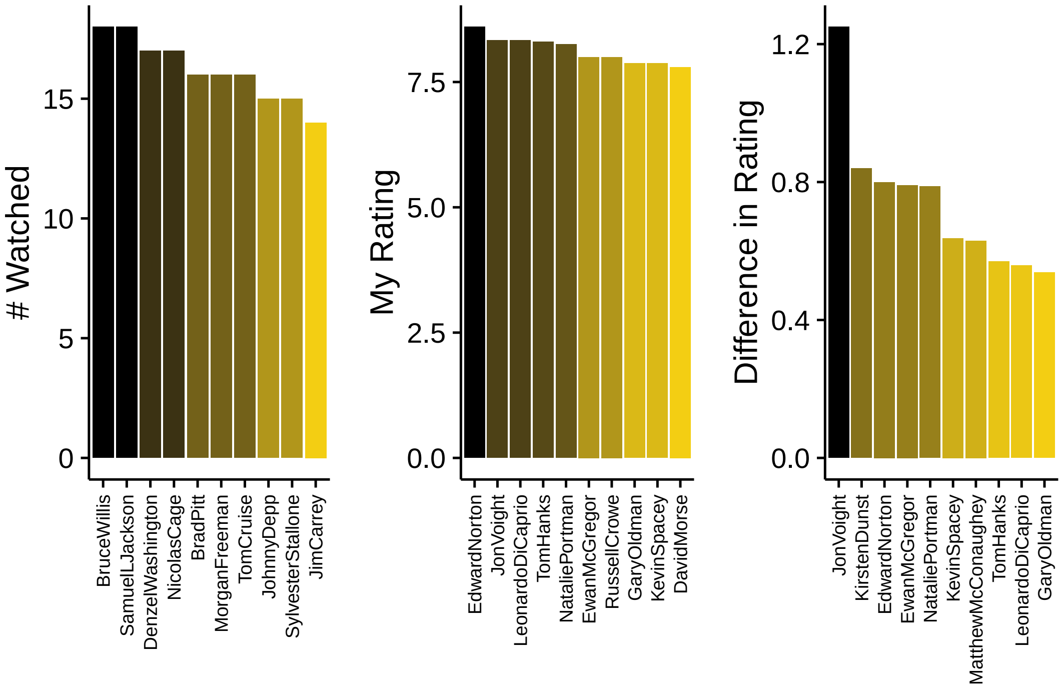
The first three bar charts display my favourite directors. The next three bar charts display my least favourite directors.

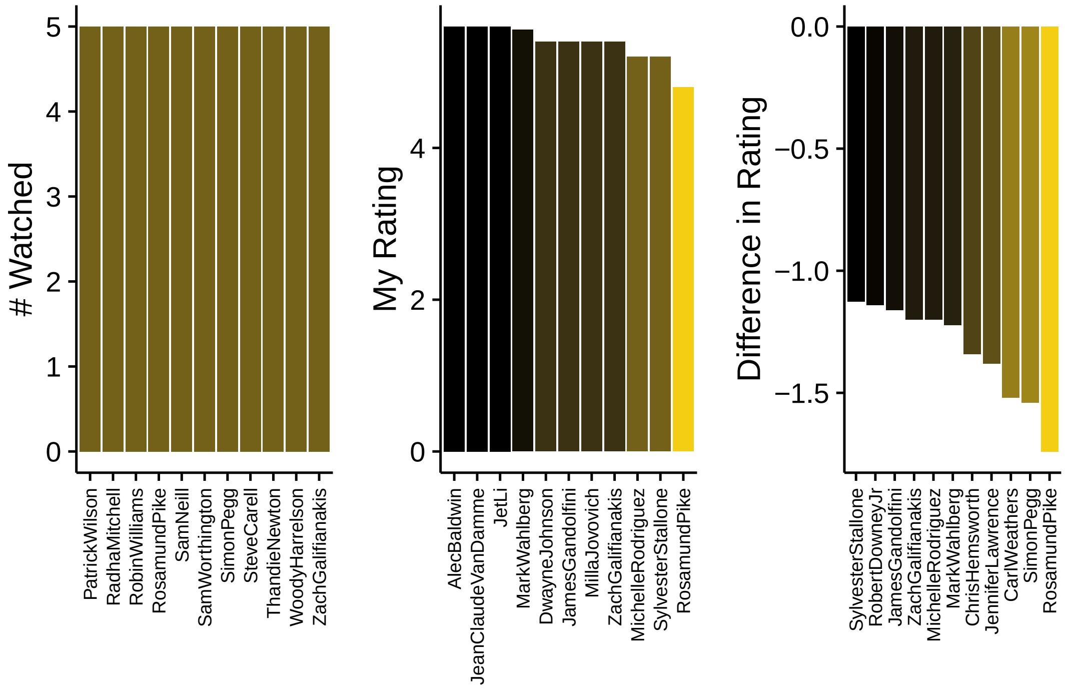




Favorite Actors

The first three bar charts display my favourite actors. The next three bar charts display my least favourite actors.





A tribute to Jon Voight

So what's up with Jon Voight? The guy hardly ever plays a lead role in a movie and he has the face of a guy who could be typecasted as a villain. So why is he in my top 10 in two out of three ranking methods? Let's see.

	Title	Yourated
8	Deliverance	9
103	Heat	10
120	Mission: Impossible	9
163	Enemy of the State	9
206	Lara Croft: Tomb Raider	6
516	National Treasure: Book of Secrets	7

No surprise here. He played a supporting role in each one of these movies I rated. Good job, Jon Voight. you sneaky bastard. This however proves a flaw in my methodology for actor score. It's bot because an actor played in a movies that I like, that I automatically appreciated all of the actors their performance. For example: I'm a big fan of 'House of Cards', but I can't stand Corey Stoll (Peter Russo). No hard feeling, Corey.

Prediction modeling

In first instance, we're splitting up the data in a training and a testing set. Furthermore, we're setting the cross-validation method to 3-fold cross validation. Raise this as you like to avoid overfitting.

```
    stripData <- function(ds = cleanData()) {
        ds <- subset(ds,select=-c(Yourated,diffRating))
        ds
    }
    createSets <- function(ds) {
        set.seed(999)
        getPackages(c("caret","caretEnsemble"))
        inTrain <- createDataPartition(y=ds$sixHigh,p=0.75,list=F)
        movieTraining <<- ds[inTrain,]
        movieTest <<- ds[-inTrain,]
        trControl <<- trainControl(method="cv",number=3)
    }
    createSets(stripData(ds))
```

Generalized Linear Model

Let's start of with the work horse of basic statistics, a generalized linear model. I use three covariates: the IMDb rating, the amount of votes the movie has received and the year the movie was released.

```
    doGLM <- function(df,testSet) {
        df <- subset(df,select=-c(Title))
        glmFit <<- train(sixHigh ~ imdbRating+imdbVotes+Year, data=df,method="glm")
        glmPredict <<- predict(glmFit,newdata=testSet)
    }
    doGLM(movieTraining,movieTest)
```

Random Forest

Next, we use a random forest model since they're capable of handling the enormous amount of dummy variables.

```
    doRF <- function(df,testSet) {
        df <- subset(df,select=-c(Title))
        rfFit <<- train(sixHigh ~.,data=df,method="rf",trainControl=trControl)
        rfPredict <<- predict(rfFit,newdata=testSet)
    }
    doRF(movieTraining,movieTest)
```

Decision Trees

Lastly, we use a decision trees model.

```
    doRPART <- function(df,testSet) {
        df <- subset(df,select=-c(Title))
        rpartFit <<- train(sixHigh ~., data=df,method="rpart")
        rpartPredict <<- predict(rpartFit,newdata=testSet)
    }
    doRPART(movieTraining,movieTest)
```

Predictions

Let's look at the accuracy of each model.

```r
modelOverview <- function(models,predictions,testSet) {
    df <- data.frame(
            confusionMatrix(predictions[[1]],testSet$sixHigh)$overall,
            confusionMatrix(predictions[[2]],testSet$sixHigh)$overall,
            confusionMatrix(predictions[[3]],testSet$sixHigh)$overall
    )
    rownames(df) <- names(confusionMatrix(predictions[[1]],testSet$sixHigh)$overall)
    colnames(df) <- c(models[[1]]$method,models[[2]]$method,models[[3]]$method)
    round(df,3)
}
kable(modelOverview(list(glmFit,rfFit,rpartFit),list(glmPredict,rfPredict,rpartPredict),movieTest))
```

	glm	rf	rpart
Accuracy	0.777	0.790	0.773
Kappa	0.552	0.576	0.544
AccuracyLower	0.719	0.733	0.715
AccuracyUpper	0.829	0.840	0.825
AccuracyNull	0.576	0.576	0.576
AccuracyPValue	0.000	0.000	0.000
McnemarPValue	0.099	0.203	0.077

Apparently, the highest accuracy we achieved is 0.79. That's not bad at all. The silly thing is that a random forest model is only slightly better than a generalized linear model. As you can see from the following graph, the reason for the only slight increase in accuracy is because of the very high explanatory power of the the IMDb Rating. Guess I suffer heavily from the herding bias.

Variable Importance

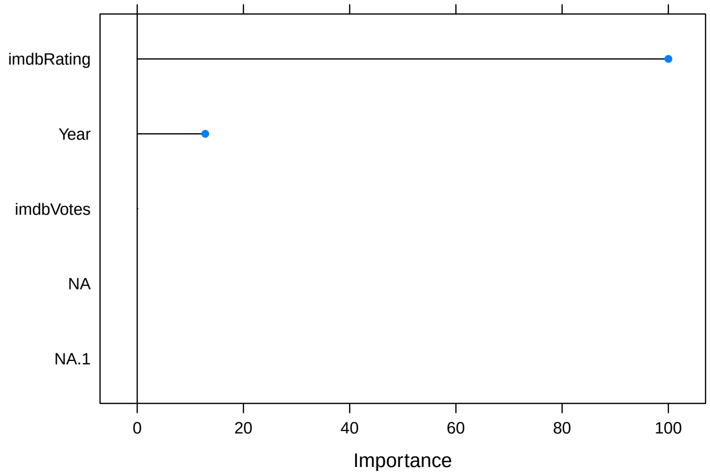


Figure 1: Variable importance in our generalized linear model.

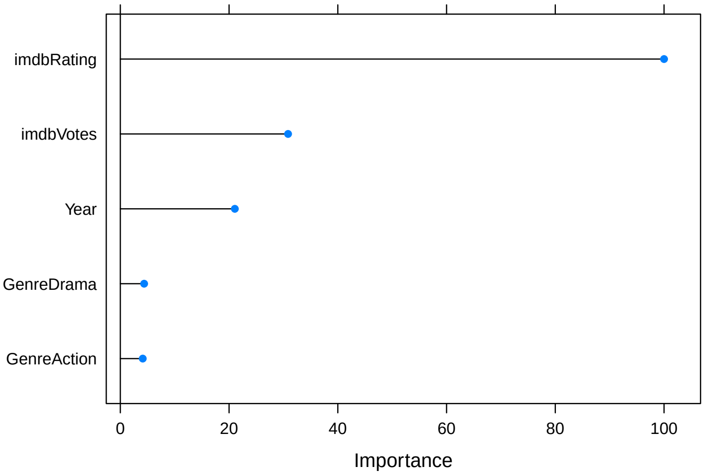


Figure 2: Variable importance in our random forest model.

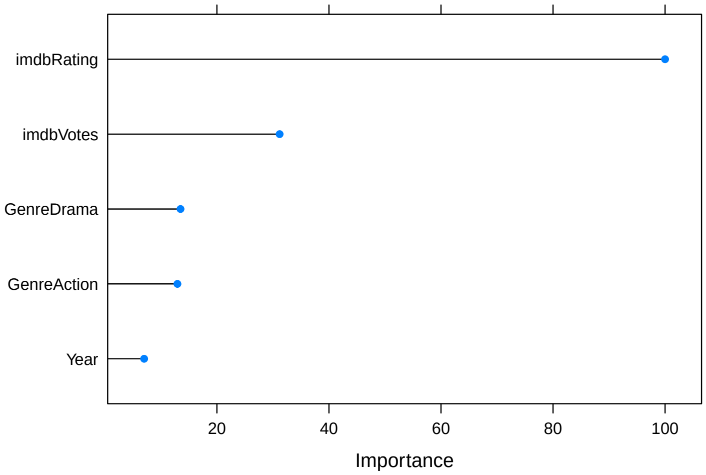


Figure 3: Variable importance in our decision tree model.