**CAT-2 PROJECT DOCUMENTATION**

**DATA MINING – 20MSSL04**

**TOPIC : IMDB SCORE PREDICTION**

**TEAM MEMBERS:**

* Dhanu Sri. B. V – 2031013
* Rithik Suthan. S – 2031035
* Saughanthikaa. K. E - 2031041

**AIM:**

Based on the massive movie information, it would be interesting to understand what are the important factors that make a movie more successful than others. So, we would like to analyze what kind of movies are more successful, in other words, get higher IMDB score. We also want to show the results of this analysis in an intuitive way by visualizing outcome using ggplot2 in R.

In this project, we take IMDB scores as response variable and focus on operating predictions by analyzing the rest of variables in the IMDB movie data. The results can help film companies to understand the secret of generating a commercial success movie.

**PROCEDURE:**

• Load the data and the relevant R libraries.

• Preprocess the data with various cleaning and recoding techniques

• Provide data visualizations of descriptive statistics of the data Fit models using statistical classification methods commonly used in churn analysis.

• Decision tree analysis , Random forest analysis , Logistic regression Examine additional data visualization of selected variables based on our modeling techniques.

**CODE AND OUTPUT:**

2.1 Load Data

{r, message=FALSE, warning=FALSE}

# Load packages

library(ggplot2) # visualization

library(ggrepel)

library(ggthemes) # visualization

library(scales) # visualization

library(dplyr) # data manipulation

library(VIM)

library(data.table)

library(formattable)

library(plotly)

library(corrplot)

library(GGally)

library(caret)

library(car)

{r}

IMDB <- read.csv("C:\\Users\\Lenovo\\OneDrive\\Desktop\\Data mining Project\\movie\_metadata.csv")

str(IMDB)

**#2.2 Remove Duplicates**

**# duplicate rows**

sum(duplicated(IMDB))

**# delete duplicate rows**

IMDB <- IMDB[!duplicated(IMDB), ]

#2.3Tidy Up Movie Title

{r, results='hide'}

library(stringr)

IMDB$movie\_title <- gsub("Â", "", as.character(factor(IMDB$movie\_title)))

str\_trim(IMDB$movie\_title, side = "right")

**OUPUT:**

[1] "Avatar"

[2] "Pirates of the Caribbean: At World's End"

[3] "Spectre"

[4] "The Dark Knight Rises"

[5] "Star Wars: Episode VII - The Force Awakens"

[6] "John Carter"

[7] "Spider-Man 3"

[8] "Tangled"

[9] "Avengers: Age of Ultron"

[10] "Harry Potter and the Half-Blood Prince"

[11] "Batman v Superman: Dawn of Justice"

[12] "Superman Returns"

[13] "Quantum of Solace"

[14] "Pirates of the Caribbean: Dead Man's Chest"

[15] "The Lone Ranger"

[16] "Man of Steel"

[17] "The Chronicles of Narnia: Prince Caspian"

[18] "The Avengers"

[19] "Pirates of the Caribbean: On Stranger Tides"

[20] "Men in Black 3"

[21] "The Hobbit: The Battle of the Five Armies"

[22] "The Amazing Spider-Man"

[23] "Robin Hood"

[24] "The Hobbit: The Desolation of Smaug"

[25] "The Golden Compass"

[26] "King Kong"

[27] "Titanic"

[28] "Captain America: Civil War"

[29] "Battleship"

………………………

………………………

………………………

[1000] "The Iron Giant"

#Split Genres

{r}

head(IMDB$genres)

OUTPUT:

[1] "Action|Adventure|Fantasy|Sci-Fi"

[2] "Action|Adventure|Fantasy"

[3] "Action|Adventure|Thriller"

[4] "Action|Thriller"

[5] "Documentary"

[6] "Action|Adventure|Sci-Fi"

# create a new data frame

genres.df <- as.data.frame(IMDB[,c("genres", "imdb\_score")])

# separate different genres into new columns

genres.df$Action <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Action") 1 else 0)

genres.df$Adventure <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Adventure") 1 else 0)

genres.df$Animation <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Animation") 1 else 0)

genres.df$Biography <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Biography") 1 else 0)

genres.df$Comedy <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Comedy") 1 else 0)

genres.df$Crime <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Crime") 1 else 0)

genres.df$Documentary <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Documentary") 1 else 0)

genres.df$Drama <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Drama") 1 else 0)

genres.df$Family <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Family") 1 else 0)

genres.df$Fantasy <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Fantasy") 1 else 0)

genres.df$`Film-Noir` <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Film-Noir") 1 else 0)

genres.df$History <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "History") 1 else 0)

genres.df$Horror <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Horror") 1 else 0)

genres.df$Musical <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Musical") 1 else 0)

genres.df$Mystery <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Mystery") 1 else 0)

genres.df$News <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "News") 1 else 0)

genres.df$Romance <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Romance") 1 else 0)

genres.df$`Sci-Fi` <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Sci-Fi") 1 else 0)

genres.df$Short <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Short") 1 else 0)

genres.df$Sport <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Sport") 1 else 0)

genres.df$Thriller <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Thriller") 1 else 0)

genres.df$War <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "War") 1 else 0)

genres.df$Western <- sapply(1:length(genres.df$genres), function(x) if (genres.df[x,1] %like% "Western") 1 else 0)

# get the mean of imdb score for different genres

means <- rep(0,23)

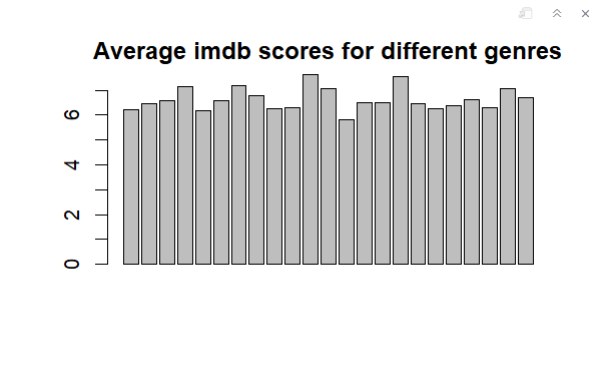
for (i in 1:23) {

means[i] <- mean(genres.df$imdb\_score[genres.df[i+2]==1])

}

# plot the means

barplot(means, main = "Average imdb scores for different genres")

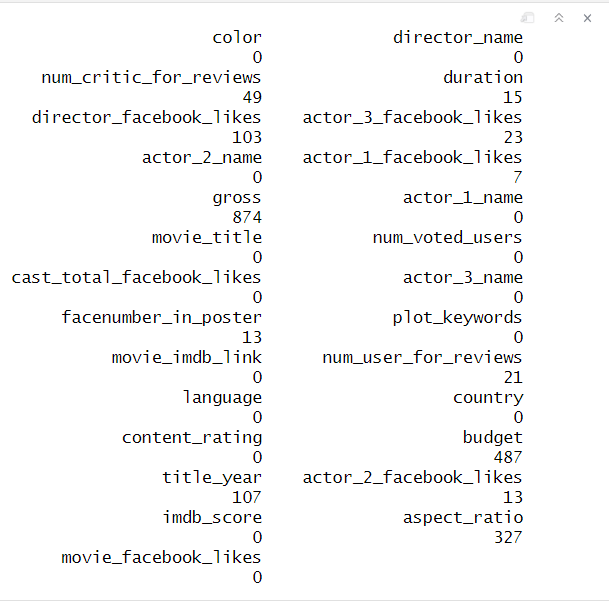


IMDB <- subset(IMDB, select = -c(genres))

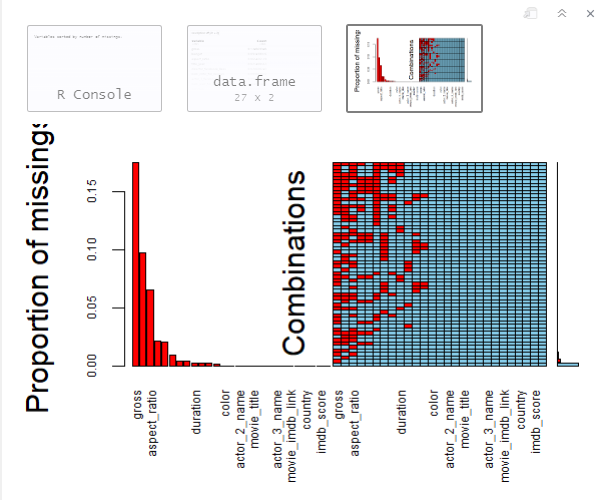
## 3 Data Cleaning

#3.1 Missing Values

colSums(sapply(IMDB, is.na))



missing.values <- aggr(IMDB, sortVars = T, prop = T, sortCombs = T, cex.lab = 1.5, cex.axis = .6, cex.numbers = 5, combined = F, gap = -.2)



#3.2 Delete some rows

IMDB <- IMDB[!is.na(IMDB$gross), ]

IMDB <- IMDB[!is.na(IMDB$budget), ]

dim(IMDB)

**[1] 3857 27**

sum(complete.cases(IMDB))

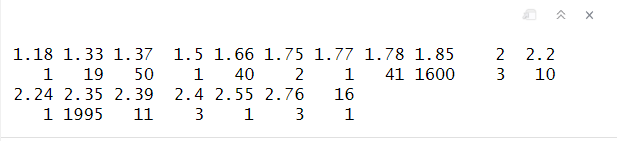
**[1] 3768**

#3.3 Analyze aspect ratio

colSums(sapply(IMDB, is.na))



table(IMDB$aspect\_ratio)



IMDB$aspect\_ratio[is.na(IMDB$aspect\_ratio)] <- 0

mean(IMDB$imdb\_score[IMDB$aspect\_ratio == 1.85])

mean(IMDB$imdb\_score[IMDB$aspect\_ratio == 2.35])

mean(IMDB$imdb\_score[IMDB$aspect\_ratio != 1.85 & IMDB$aspect\_ratio != 2.35])

[1] 6.373938

[1] 6.508471

[1] 6.672519

IMDB <- subset(IMDB, select = -c(aspect\_ratio))

# replace NA with column average for facenumber\_in\_poster

IMDB$facenumber\_in\_poster[is.na(IMDB$facenumber\_in\_poster)] <- round(mean(IMDB$facenumber\_in\_poster, na.rm = TRUE))

# convert 0s into NAs for other predictors

IMDB[,c(5,6,8,13,24,26)][IMDB[,c(5,6,8,13,24,26)] == 0] <- NA

# impute missing value with column mean

IMDB$num\_critic\_for\_reviews[is.na(IMDB$num\_critic\_for\_reviews)] <- round(mean(IMDB$num\_critic\_for\_reviews, na.rm = TRUE))

IMDB$duration[is.na(IMDB$duration)] <- round(mean(IMDB$duration, na.rm = TRUE))

IMDB$director\_facebook\_likes[is.na(IMDB$director\_facebook\_likes)] <- round(mean(IMDB$director\_facebook\_likes, na.rm = TRUE))

IMDB$actor\_3\_facebook\_likes[is.na(IMDB$actor\_3\_facebook\_likes)] <- round(mean(IMDB$actor\_3\_facebook\_likes, na.rm = TRUE))

IMDB$actor\_1\_facebook\_likes[is.na(IMDB$actor\_1\_facebook\_likes)] <- round(mean(IMDB$actor\_1\_facebook\_likes, na.rm = TRUE))

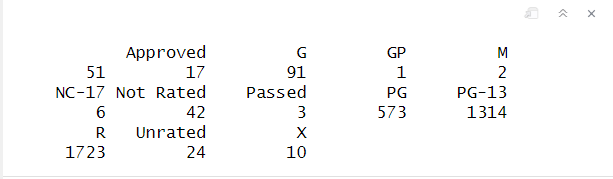
IMDB$cast\_total\_facebook\_likes[is.na(IMDB$cast\_total\_facebook\_likes)] <- round(mean(IMDB$cast\_total\_facebook\_likes, na.rm = TRUE))

IMDB$actor\_2\_facebook\_likes[is.na(IMDB$actor\_2\_facebook\_likes)] <- round(mean(IMDB$actor\_2\_facebook\_likes, na.rm = TRUE))

IMDB$movie\_facebook\_likes[is.na(IMDB$movie\_facebook\_likes)] <- round(mean(IMDB$movie\_facebook\_likes, na.rm = TRUE))

#3.5 Sort out content ratings

table(IMDB$content\_rating)



IMDB <- IMDB[!(IMDB$content\_rating %in% ""),]

IMDB$content\_rating[IMDB$content\_rating == 'M'] <- 'PG'

IMDB$content\_rating[IMDB$content\_rating == 'GP'] <- 'PG'

IMDB$content\_rating[IMDB$content\_rating == 'X'] <- 'NC-17'

IMDB$content\_rating[IMDB$content\_rating == 'Approved'] <- 'R'

IMDB$content\_rating[IMDB$content\_rating == 'Not Rated'] <- 'R'

IMDB$content\_rating[IMDB$content\_rating == 'Passed'] <- 'R'

IMDB$content\_rating[IMDB$content\_rating == 'Unrated'] <- 'R'

IMDB$content\_rating <- factor(IMDB$content\_rating)

table(IMDB$content\_rating)

IMDB <- IMDB %>%

mutate(profit = gross - budget,

return\_on\_investment\_perc = (profit/budget)\*100)

table(IMDB$color)

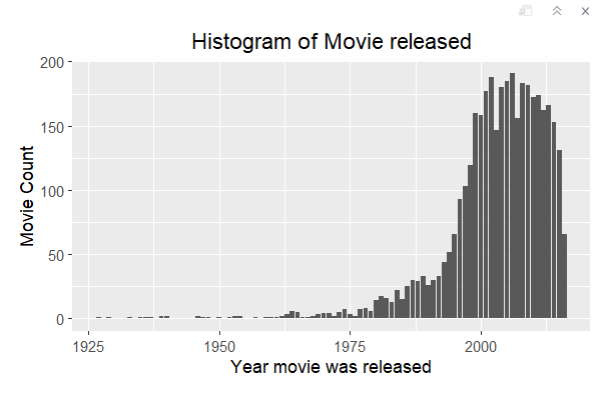
IMDB <- subset(IMDB, select = -c(color))

ggplot(IMDB, aes(title\_year)) +

geom\_bar() +

labs(x = "Year movie was released", y = "Movie Count", title = "Histogram of Movie released") +

theme(plot.title = element\_text(hjust = 0.5))



###Top 20 movies based on its Profit

IMDB %>%

filter(title\_year %in% c(2000:2016)) %>%

arrange(desc(profit)) %>%

top\_n(20, profit) %>%

ggplot(aes(x=budget/1000000, y=profit/1000000)) +

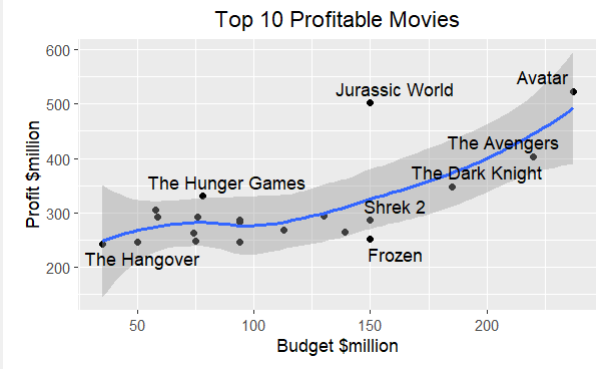
geom\_point() +

geom\_smooth() +

geom\_text\_repel(aes(label=movie\_title)) +

labs(x = "Budget $million", y = "Profit $million", title = "Top 10 Profitable Movies") +

theme(plot.title = element\_text(hjust = 0.5))



## Top 20 movies based on its Return on Investment

IMDB %>%

filter(budget > 100000) %>%

mutate(profit = gross - budget,

return\_on\_investment\_perc = (profit/budget)\*100) %>%

arrange(desc(profit)) %>%

top\_n(20, profit) %>%

ggplot(aes(x=budget/1000000, y = return\_on\_investment\_perc)) +

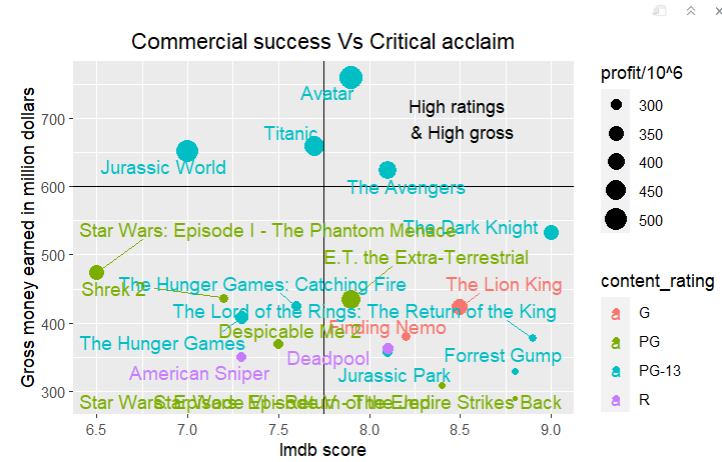
geom\_point(size = 2) +

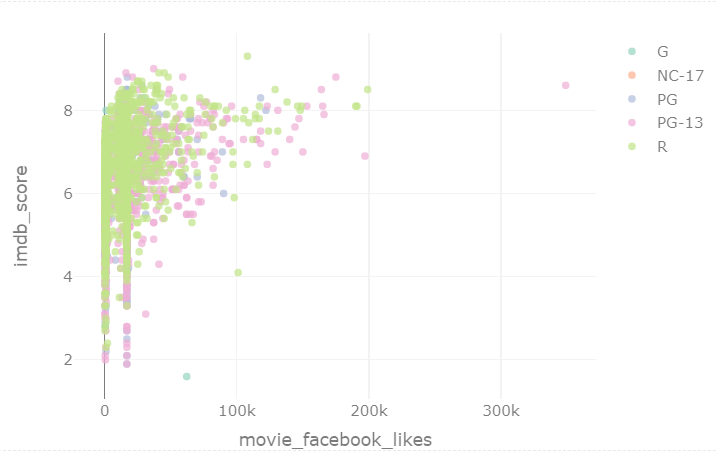
geom\_smooth(size = 1) +

geom\_text\_repel(aes(label = movie\_title), size = 3) +

xlab("Budget $million") +

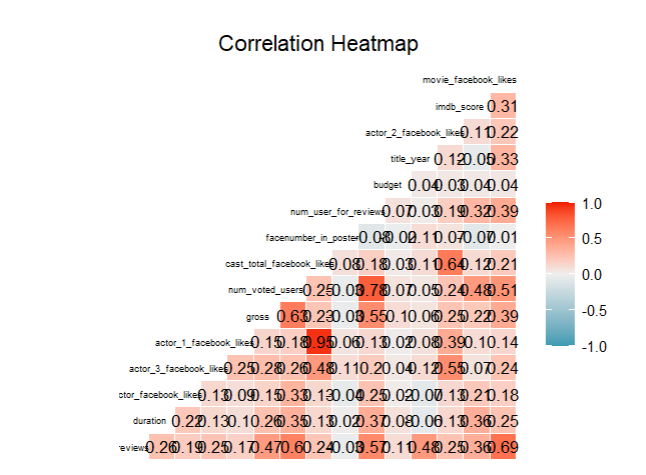
ylab("Percent Return on Investment") +

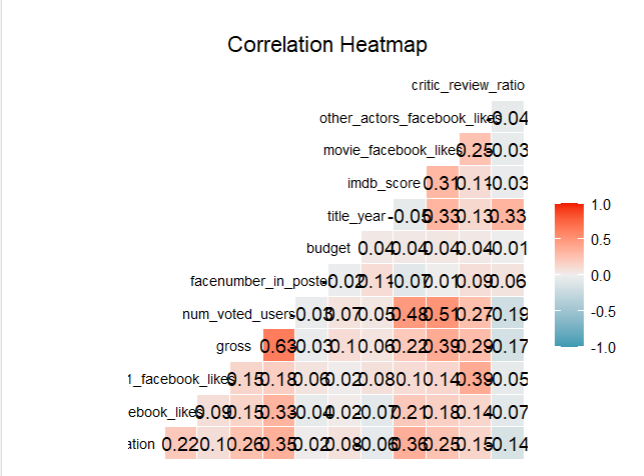
ggtitle("20 Most Profitable Movies based on its Return on Investment")

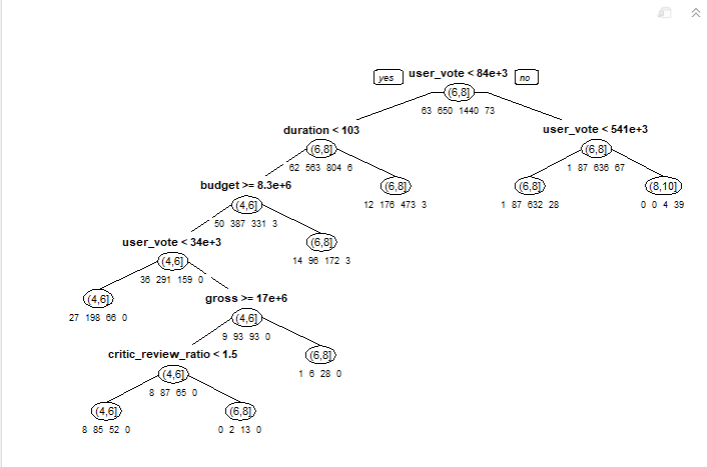


[1] 1660

[1] 3621







Classification tree:

rpart(formula = binned\_score ~ . - imdb\_score, data = train,

method = "class", cp = 1e-05, minsplit = 5, xval = 5)

Variables actually used in tree construction:

[1] actor1\_fb budget content

[4] country critic\_review\_ratio director\_fb

[7] duration face\_number gross

[10] movie\_fb other\_actors\_fb user\_vote

[13] year

Root node error: 786/2226 = 0.3531

n= 2226

CP nsplit rel error xerror xstd

1 0.05597964 0 1.00000 1.00000 0.028688

2 0.04452926 3 0.83206 0.91730 0.028090

3 0.01399491 4 0.78753 0.85751 0.027580

4 0.00890585 7 0.74555 0.81679 0.027193

5 0.00805768 8 0.73664 0.80153 0.027040

6 0.00636132 11 0.71247 0.79517 0.026974

7 0.00572519 13 0.69975 0.78753 0.026895

8 0.00559796 17 0.67684 0.78753 0.026895

9 0.00508906 28 0.60051 0.78880 0.026908

10 0.00445293 32 0.58015 0.81552 0.027181

11 0.00413486 34 0.57125 0.80916 0.027117

12 0.00381679 38 0.55471 0.82061 0.027231

13 0.00349873 43 0.53435 0.82824 0.027305

14 0.00339271 47 0.52036 0.82570 0.027281

15 0.00318066 53 0.50000 0.82570 0.027281

16 0.00254453 67 0.45038 0.83461 0.027366

17 0.00218103 103 0.35623 0.86005 0.027603

18 0.00190840 113 0.32952 0.86514 0.027649

19 0.00169635 131 0.29389 0.87659 0.027750

20 0.00127226 145 0.26590 0.93003 0.028190

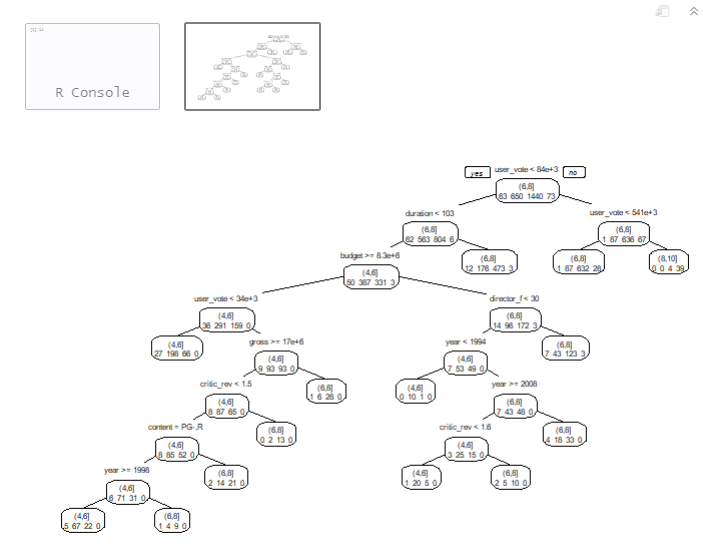
21 0.00095420 207 0.18702 0.93257 0.028210

22 0.00076336 219 0.17176 0.93766 0.028249

23 0.00063613 224 0.16794 0.94148 0.028277

24 0.00042409 234 0.16158 0.94784 0.028325

25 0.00001000 237 0.16031 0.95165 0.028353



Confusion Matrix and Statistics

Reference

Prediction (0,4] (4,6] (6,8] (8,10]

(0,4] 0 0 0 0

(4,6] 33 295 94 0

(6,8] 30 355 1342 34

(8,10] 0 0 4 39

Overall Statistics

Accuracy : 0.7529

95% CI : (0.7345, 0.7707)

No Information Rate : 0.6469

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.4284

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: (0,4] Class: (4,6] Class: (6,8]

Sensitivity 0.0000 0.4538 0.9319

Specificity 1.0000 0.9194 0.4669

Pos Pred Value NaN 0.6991 0.7621

Neg Pred Value 0.9717 0.8032 0.7892

Prevalence 0.0283 0.2920 0.6469

Detection Rate 0.0000 0.1325 0.6029

Detection Prevalence 0.0000 0.1896 0.7911

Balanced Accuracy 0.5000 0.6866 0.6994

Class: (8,10]

Sensitivity 0.53425

Specificity 0.99814

Pos Pred Value 0.90698

Neg Pred Value 0.98443

Prevalence 0.03279

Detection Rate 0.01752

Detection Prevalence 0.01932

Balanced Accuracy 0.76619

Confusion Matrix and Statistics

Reference

Prediction (0,4] (4,6] (6,8] (8,10]

(0,4] 0 0 0 0

(4,6] 13 61 41 0

(6,8] 9 121 465 16

(8,10] 0 0 5 11

Overall Statistics

Accuracy : 0.7237

95% CI : (0.69, 0.7556)

No Information Rate : 0.6887

P-Value [Acc > NIR] : 0.02076

Kappa : 0.299

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: (0,4] Class: (4,6] Class: (6,8]

Sensitivity 0.00000 0.33516 0.9100

Specificity 1.00000 0.90357 0.3680

Pos Pred Value NaN 0.53043 0.7610

Neg Pred Value 0.97035 0.80702 0.6489

Prevalence 0.02965 0.24528 0.6887

Detection Rate 0.00000 0.08221 0.6267

Detection Prevalence 0.00000 0.15499 0.8235

Balanced Accuracy 0.50000 0.61937 0.6390

Class: (8,10]

Sensitivity 0.40741

Specificity 0.99301

Pos Pred Value 0.68750

Neg Pred Value 0.97796

Prevalence 0.03639

Detection Rate 0.01482

Detection Prevalence 0.02156

Balanced Accuracy 0.70021

Confusion Matrix and Statistics

Reference

Prediction (0,4] (4,6] (6,8] (8,10]

(0,4] 0 0 0 0

(4,6] 4 82 35 0

(6,8] 6 151 429 17

(8,10] 0 0 4 15

Overall Statistics

Accuracy : 0.7079

95% CI : (0.6738, 0.7404)

No Information Rate : 0.6299

P-Value [Acc > NIR] : 4.486e-06

Kappa : 0.3311

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: (0,4] Class: (4,6] Class: (6,8]

Sensitivity 0.00000 0.3519 0.9167

Specificity 1.00000 0.9235 0.3673

Pos Pred Value NaN 0.6777 0.7114

Neg Pred Value 0.98654 0.7572 0.7214

Prevalence 0.01346 0.3136 0.6299

Detection Rate 0.00000 0.1104 0.5774

Detection Prevalence 0.00000 0.1629 0.8116

Balanced Accuracy 0.50000 0.6377 0.6420

Class: (8,10]

Sensitivity 0.46875

Specificity 0.99437

Pos Pred Value 0.78947

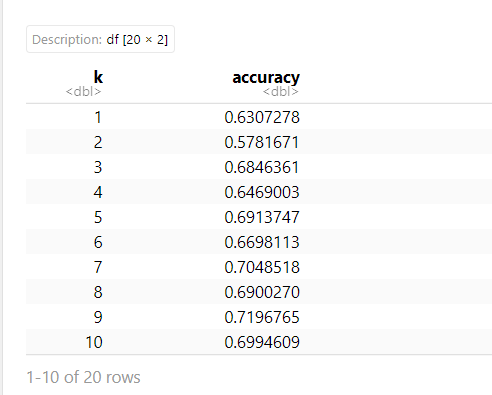
Neg Pred Value 0.97652

Prevalence 0.04307

Detection Rate 0.02019

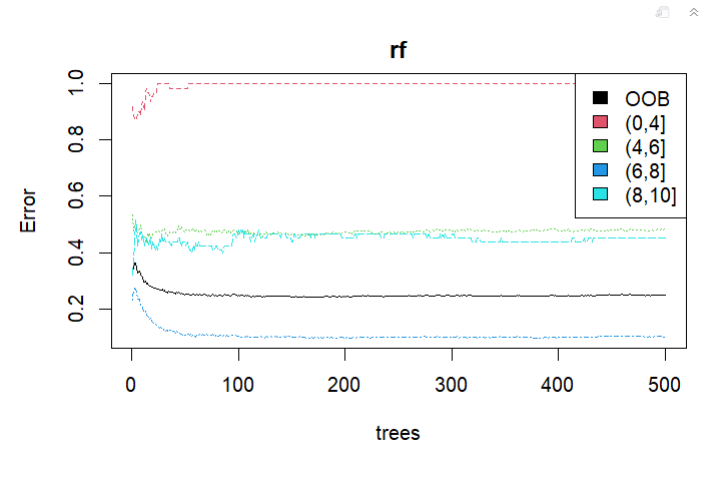
Detection Prevalence 0.02557

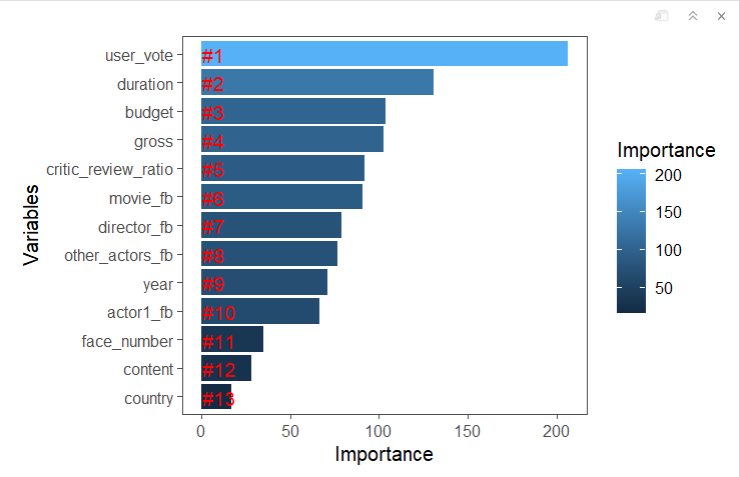
Balanced Accuracy 0.73156



Accuracy

0.6864065





Confusion Matrix and Statistics

Reference

Prediction (0,4] (4,6] (6,8] (8,10]

(0,4] 0 0 0 0

(4,6] 16 98 40 0

(6,8] 6 84 470 15

(8,10] 0 0 1 12

Overall Statistics

Accuracy : 0.7817

95% CI : (0.7502, 0.8109)

No Information Rate : 0.6887

P-Value [Acc > NIR] : 1.041e-08

Kappa : 0.4736

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: (0,4] Class: (4,6] Class: (6,8]

Sensitivity 0.00000 0.5385 0.9198

Specificity 1.00000 0.9000 0.5455

Pos Pred Value NaN 0.6364 0.8174

Neg Pred Value 0.97035 0.8571 0.7545

Prevalence 0.02965 0.2453 0.6887

Detection Rate 0.00000 0.1321 0.6334

Detection Prevalence 0.00000 0.2075 0.7749

Balanced Accuracy 0.50000 0.7192 0.7326

Class: (8,10]

Sensitivity 0.44444

Specificity 0.99860

Pos Pred Value 0.92308

Neg Pred Value 0.97942

Prevalence 0.03639

Detection Rate 0.01617

Detection Prevalence 0.01752

Balanced Accuracy 0.72152

Confusion Matrix and Statistics

Reference

Prediction (0,4] (4,6] (6,8] (8,10]

(0,4] 0 0 0 0

(4,6] 7 117 43 0

(6,8] 3 116 421 14

(8,10] 0 0 4 18

Overall Statistics

Accuracy : 0.7483

95% CI : (0.7155, 0.7792)

No Information Rate : 0.6299

P-Value [Acc > NIR] : 4.04e-12

Kappa : 0.4512

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: (0,4] Class: (4,6] Class: (6,8]

Sensitivity 0.00000 0.5021 0.8996

Specificity 1.00000 0.9020 0.5164

Pos Pred Value NaN 0.7006 0.7599

Neg Pred Value 0.98654 0.7986 0.7513

Prevalence 0.01346 0.3136 0.6299

Detection Rate 0.00000 0.1575 0.5666

Detection Prevalence 0.00000 0.2248 0.7456

Balanced Accuracy 0.50000 0.7021 0.7080

Class: (8,10]

Sensitivity 0.56250

Specificity 0.99437

Pos Pred Value 0.81818

Neg Pred Value 0.98058

Prevalence 0.04307

Detection Rate 0.02423

Detection Prevalence 0.02961

Balanced Accuracy 0.77844

CLUSTERING :

# Load the required libraries

library(ggplot2)

library(dplyr)

library(factoextra)

# Load the dataset

movies <- read.csv("C:\\Users\\Lenovo\\OneDrive\\Desktop\\Data mining Project\\movie\_metadata.csv",stringsAsFactors = FALSE)

# Data cleaning

movies\_clean <- movies %>%

select(num\_critic\_for\_reviews, duration, director\_facebook\_likes,

actor\_3\_facebook\_likes, actor\_1\_facebook\_likes, gross,

num\_voted\_users, cast\_total\_facebook\_likes, num\_user\_for\_reviews,

budget, title\_year, imdb\_score, movie\_facebook\_likes) %>%

na.omit()

# Feature scaling

movies\_scaled <- scale(movies\_clean)

# Elbow method to determine the optimal number of clusters

set.seed(123)

fviz\_nbclust(movies\_scaled, kmeans, method = "wss") +

theme\_classic() + ggtitle("Elbow Method")

# K-means clustering

set.seed(123)

kmeans\_model <- kmeans(movies\_scaled, centers = 4, nstart = 25)

movies\_clustered <- movies\_clean %>%

mutate(cluster = kmeans\_model$cluster)

# Cluster analysis

movies\_cluster\_analysis <- movies\_clustered %>%

group\_by(cluster) %>%

summarize(

mean\_num\_critic\_for\_reviews = mean(num\_critic\_for\_reviews),

mean\_duration = mean(duration),

mean\_director\_facebook\_likes = mean(director\_facebook\_likes),

mean\_actor\_3\_facebook\_likes = mean(actor\_3\_facebook\_likes),

mean\_actor\_1\_facebook\_likes = mean(actor\_1\_facebook\_likes),

mean\_gross = mean(gross),

mean\_num\_voted\_users = mean(num\_voted\_users),

mean\_cast\_total\_facebook\_likes = mean(cast\_total\_facebook\_likes),

mean\_num\_user\_for\_reviews = mean(num\_user\_for\_reviews),

mean\_budget = mean(budget),

mean\_title\_year = mean(title\_year),

mean\_imdb\_score = mean(imdb\_score),

mean\_movie\_facebook\_likes = mean(movie\_facebook\_likes)

)

# Visualize the clusters

ggplot(movies\_clustered, aes(x = imdb\_score, y = gross, color = factor(cluster))) +

geom\_point(alpha = 0.8, size = 3) +

scale\_color\_discrete(name = "Cluster") +

theme\_classic() +

labs(title = "K-Means Clustering",

x = "IMDB Score",

y = "Gross") +

theme(legend.position = "bottom")

