IST707 Final Project

Zachary Chipman

6/21/2021

## Introduction

Rotten Tomatoes is a website where movies and TV shows are rated as either fresh or rotten based on the percentage of positive reviews it receives. If more than 60% of the critics wrote a positive review for a film, then it is rated fresh. Any less, and it’s rated rotten. Films can also be rated as “Certified Fresh” if a certain number of what Rotten Tomatoes calls “Top Critics” give the film a positive review.

Websites like Rotten Tomatoes give analysts the opportunity to see what factors determine the critical reception of the film. Machine learning can be used to not only see what attributes lead to a positive or negative review but can also be used to predict critical reception. This project will explore a Kaggle data set containing the characteristics of a number of films as well as the critical and audience reception from Rotten Tomatoes. Analysis will also be done for audience opinion so that it may compared to the critic analysis.

## Analysis and Models

Note: Only important output/plots of the code will be displayed in the project proper. The full code with all outputs and plots will be included in the appendix of this paper.

### About the Data

The unaltered data set had 17,712 observations of 22 variables, some of which were removed during cleaning. Columns like the film’s title were removed because they did not provide any relevant information for analysis while others like Director were removed because there were thousands of unique observations that would inhibit the generalization of the data.

There were also some columns that were removed because they would hurt the integrity of the analysis. These included the actual percentage of positive reviews, known as CRating and ARating. A machine learning model would only have to look at these values to determine the status of the film. Since the goal of this experiment was to see what other factors might determine a film’s reception, these columns were removed.

Two text-based variables, Info and Consensus, were removed because they were out of this project’s scope. These columns are a brief summary and the critical consensus of the film respectively and may be used in future research.

Below are descriptions of the variables that were used as well as the details of how they were cleaned for analysis:

CStatus/AStatus (converted to factor): These are the variables that this paper will attempt to predict. CStatus can be either Certified Fresh, Fresh, or Rotten. AStatus can be either Upright or Spilled.

Content (converted to factor): The content rating that the Motion Picture Association gave the film due to its content. Every film in the data set had one of the following ratings: G, PG, PG-13, R, or NR (Not Rated). It should be noted that some of the films in the original data set were given the NC-17 rating but were removed due to a lack of observations.

Genre (converted to factor): Originally, the films could have multiple genres so the column was manipulated so only the first genre listed would be used. Genres that had less than 100 observations were removed to make running the models easier. The final 9 genres were: Action & Adventure, Animation, Art House & International, Classics, Comedy, Documentary, Drama, Mystery & Suspense, and Horror.

Release (Date): The release date to the film. From this variable, three other columns were created. The first was “Rdecade,” which was the year of release binned by decade. The second was “Rmonth,” which was the month the film was released. Finally, there was “Rmonth,” which was the time of the month the film was released (either Early, Mid, or Late Month). All of these added columns were converted into factors.

Stream (Date): The date that the film was released on streaming services. This variable was also used to make three different columns that were like the ones made from the “Release” column. The one exception was that the stream year, “Syear,” was not binned by the decade but rather by year (as the earliest year in the data was 1998).

Runtime (Date): The run time, in minutes, of the film. For the purpose of analysis, this column was binned by 1-hour bins and converted into an ordered factor.

After NAs were removed and all other data cleaning was conducted, the data set had 15,814 observations of 13 variables.

For the sake of brevity, only the graphs that show actionable insight into the data will be included into the paper. Charts for all the variables can be found in the appendix. Each graph will be created twice, once filled with CStatus and then again with AStatus for a side-by-side comparison. The color palate for these plots comes from a package called “wesanderson.” Wes Anderson is a director whose films have a unique style all their own. This palette is inspired by his film “The Darjeeling Limited.”

Chart, bar chart

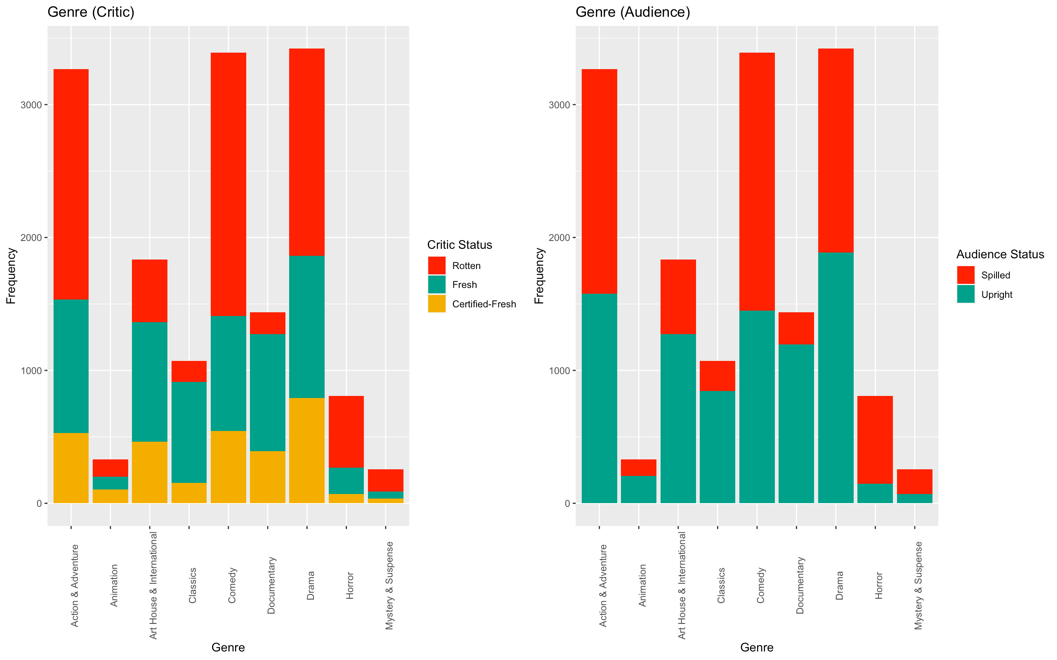
Description automatically generated

This graph shows that more than half of the films were rated positively (by both critics and audience members). A general agreement can also be seen between the two groups. For example, most of the Films rated fresh by critics were rated upright by audiences.

Chart, bar chart

Description automatically generated

Most of the films are either rated R or NR.



Most of the films are either an Action & Adventure, Comedy, or a Drama. The smaller genres seem to obtain more favorable ratings from both critics and audiences (Horror and Mystery being the exceptions).

Chart, bar chart, waterfall chart

Description automatically generated

As expected, the number of films in the data set increases by decade. The higher proportion of Fresh to Rotten films in earlier decades makes sense since most of those films are considered classics and are more likely to be rated favorably.

Chart, bar chart

Description automatically generated

One thing to note in these charts is that more films are released in January than any other month. This may be due because producers and directors want to get theirs films noticed in time for award season. There is also surprisingly no spike in the summer months when all the major “blockbusters” usually come out.

Chart, bar chart, waterfall chart

Description automatically generated

Most of the films are between 1-2 hours in length with some falling between 2-3 hours.

### Models

Two modeling methods will be used in this section, Association Rules and Classification. The former will be used to find out what characteristics make films more lauded by critics and/or audiences while the latter will be used to find a model that will most accurately predict how a future film will be received. Each of the models will be run twice, once for critical reception and again for audience reception.

In both cases of the Association Rules mining, continuous variables are removed to aid the analysis. The minimum confidence was set at 0.6 while the support was tweaked until 20-30 strong rules were created. Below is a summary of the rules mining. The code, as well as all the rules generated, can be found in the appendix.

#### Critic Association Rules

Critic Status = Fresh: 23 Rules Generated, Support ranges from 0.03 to 0.17, Confidence ranges from 0.6 to 0.75, Lift ranges from 1.64 to 2.02.

Critic Status = Rotten: 25 Rules Generated, Support ranges from 0.03 to 0.09, Confidence ranges from 0.6 to 0.71, Lift ranges from 1.38 to 1.62.

#### Audience Association Rules

Audience Status = Upright: 23 Rules Generated, Support ranges from 0.05 to 0.18, Confidence ranges from 0.6 to 0.83, Lift ranges from 1.10 to 1.52.

Audience Status = Spilled: 26 Rules Generated, Support ranges from 0.03 to 0.08, Confidence ranges from 0.6 to 0.83, Lift ranges from 1.33 to 1.84.

### Classification Models

The data was split using the two-thirds rule, with 2/3 of the data being part of the training data while the remaining 1/3 was used as the testing data. Some of of the columns were also removed to avoid redundancy and help the models run faster. The release and run time variables were removed in favor of their binned counterparts. The variables related to streaming dates were also removed because, although they appeared in the association rules, there was no trend in the rules that signified a relation between them and a film’s reception. Finally, it must be noted that the Critic/Audience Status column was removed from the test data and stored in a separate vector called “ctestanswers” and “atestanswers” respectively. These vectors will be used in a confusion matrix to test the prediction models.

Caret’s “trainControl” function will be used to train each of the models. The models will undergo 5-fold cross validation five times with a certain parameter being tuned each time. The most accurate of these models will then be used for analysis. As the function is a random function, the “set.seed” function was used to ensure reproducibility. Four training methods will be used: Decision Tree, Support Vector Machines (SVM), k-Nearest Neignbor (kNN), and Random Forest. The tuning parameters and the resulting accuracy are summarized in the following paragraphs. More detail can be found in the appendix.

The decision tree model is tuned be the complexity parameter (cp), which controls the size of the tree. The final value used for the model was cp = 0.004389667, resulting in an accuracy of 0.58.

In SVM models, kernels are used to classify linearly inseparable data sets. This model uses a radial kernel and is tuned by the cost parameter C, which is the penalty for an inaccurate prediction. The final value used for the model was C = 4, resulting in an accuracy of 0.58.

kNN is tuned by the the number of “nearest neighbors” that are considered during training (k). The final value used for the model was k = 13, resulting in an accuracy of 0.56.

Random Forest is tuned using “mtry.” The Rdocumentation defines this parameter as the “number of variables randomly sampled as candidates at each split.” This parameter has a more significant affect on the accuracy than the number of trees does, which is set to 500 by default. The final value used for the model was mtry = 2, resulting in an accuracy of 0.57.

Chart, scatter chart

Description automatically generated

By plotting the accuracy of each model together, the SVM and decision tree models are the most accurate.

When these models are applied to the test data, the accuracy is not much higher (around 0.58-0.59). The results for all the confusion matrices can be found in the appendix.

#### Audience Classification Models

Chart

Description automatically generated

The same setup and parameters were used to predict the audience status. The decision tree had a cp = 0.005426357 and an accuracy of 0.67. The SVM had a C = 4 and an accuracy of 0.67. The kNN had a k = 13 and an accuracy of 0.65. Finally, the random forest had a mtry = 2 and an accuracy of 0.66. Like the CStatus models, the SVM and decision tree were slightly more accurate and none of the models saw a significant increase in accuracy when applied to the test data (0.65-0.67). However, the AStatus accuracy was higher for both the train and test data sets. The possible implications of this will be discussed in the following sections.

## Results

### Association Rules Mining

One trend that is hard to deny is that both critics and audiences favored films that were not rated. Critics and audiences rated these films positively over 60% of the time, with 17/23 and 16/23 rules respectively having Content = NR on the left-hand side. Both groups loved documentaries, with audiences rating them upright a whopping 83% of the time. Classics were also, unsurprisingly, highly favored, with a 71% confidence from critics and a 79% confidence from audiences.

Chart, bar chart

Description automatically generated

This chart gives indication that there may be some overlap in these observations as many classics and documentaries are not rated.

The love of classics is self-explanatory, but the love of documentaries might be since they are non-fiction, and therefore more real. It is not hard to understand or emphasize with the characters in a documentary because their experiences and struggles are genuine.

Comedy and Horror movies were the most unfavorably rated genres among both groups, with horror being especially panned by audiences (Spilled = 82% confidence). Horror films tend to have a reputation of being low budget and relying on cheap gimmicks (i.e., jump scares) to frighten their audiences, which may explain this result.

Films rated PG-13 were not well received by critics, with 13/25 of the rules having this characteristic. The rating of PG-13 was not introduced until the early 1980’s, which may suggest that critics are less inclined to enjoy modern films. However, other rules suggest that audiences may be even more critical of modern films as 16/26 of the rules include films that were released in the 2010’s.

### Classification Models

The accuracy of the models was not extremely accurate. Both the train and test predictions were around 57-58% accurate regarding critic behavior while they were slightly more accurate for audience behavior (65-67%). One theory is that while the status of critics could be one of three options the audience status could only one of two, therefore having less chance of an incorrect prediction. It is also difficult to say what model works the best as the accuracy of the models were so close together.

## Conclusions

This project was a good start towards the goal of using machine learning to predict critic and audience behavior. However, time affected the scope and detail of the project so there are several steps that will be conducted in future research.

The association rules for the critic status may have been affected by the fact that Certified Fresh films were not considered. In further study, Fresh and Certified Fresh films will be combined to hopefully obtain higher levels of support and confidence.

Actions would be taken to improve the accuracy of the models. One of the issues may have been that there were too many genres in the data set for accurate classification to be conducted. If the focus was on what factors made a certain genre more favorable to critics and audiences, then a more accurate model may have been obtained. The stream-related variables as well as classic films should also be removed. After all, saying that classics are well received is a dull and unactionable insight.

Finally, as stated earlier, the text variables (Info and Consensus) were removed because text mining was out of the scope of this project. But performing sentiment analysis (as well as other machine learning techniques) on these columns may provide insights that this project did not find.

## Appendix

# Libraries required for this project  
library(plyr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
library(readr)  
library(caret)

## Loading required package: lattice

library(caretEnsemble)

##   
## Attaching package: 'caretEnsemble'

## The following object is masked from 'package:ggplot2':  
##   
## autoplot

library(questionr)  
library(klaR)

## Loading required package: MASS

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

library(e1071)   
library(rpart)   
library(rpart.plot)   
library(rattle)

## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(imager)

## Loading required package: magrittr

##   
## Attaching package: 'imager'

## The following object is masked from 'package:magrittr':  
##   
## add

## The following object is masked from 'package:plyr':  
##   
## liply

## The following objects are masked from 'package:stats':  
##   
## convolve, spectrum

## The following object is masked from 'package:graphics':  
##   
## frame

## The following object is masked from 'package:base':  
##   
## save.image

library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:imager':  
##   
## grow

## The following object is masked from 'package:rattle':  
##   
## importance

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following object is masked from 'package:dplyr':  
##   
## recode

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

library(arulesViz)  
library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:randomForest':  
##   
## combine

## The following object is masked from 'package:dplyr':  
##   
## combine

library(wesanderson)  
  
# Read in the data set (make sure that the cwd is correct)  
movies <- read.csv("rotten\_tomatoes\_movies.csv", header = T,   
 na.strings = c("", "NA"))  
OGmovies <- movies  
  
# Data Cleaning ############################################  
  
# Rename Columns  
movies <- plyr::rename(movies, c("rotten\_tomatoes\_link"="Link",   
 "movie\_title"="Title",  
 "movie\_info"="Info",  
 "critics\_consensus"="Consensus",  
 "content\_rating"="Content",  
 "genres"="Genre",  
 "directors"="Director",  
 "authors"="Writer",  
 "actors"="Actor",  
 "original\_release\_date"="Release",  
 "streaming\_release\_date"="Stream",  
 "runtime"="Runtime",  
 "production\_company"="Production",  
 "tomatometer\_status"="CStatus",  
 "tomatometer\_rating"="CRating",  
 "tomatometer\_count"="CCount",  
 "audience\_status"="AStatus",  
 "audience\_rating"="ARating",  
 "audience\_count"="ACount",  
 "tomatometer\_top\_critics\_count"="Top",  
 "tomatometer\_fresh\_critics\_count"="Fresh",  
 "tomatometer\_rotten\_critics\_count"="Rotten"))  
   
# Remove all columns that are not needed for analysis.  
movies <- subset(movies, select = -c(Link, Title, Director, Writer, Actor,   
 Production, CRating, CCount, ARating, ACount, Top, Fresh,   
 Rotten))  
  
# Modify the genre column so that it only shows the first genre listed.  
movierows <- nrow(movies)  
for (x in 1:movierows) {  
 movies$Genre[x] <- gsub("(.\*?),.\*", "\\1", movies$Genre[x])  
}  
  
# Covert nominal variables into factors  
movies$Content = factor(movies$Content)  
movies$Genre = factor(movies$Genre)  
movies$CStatus = ordered(movies$CStatus, levels = c("Rotten", "Fresh",   
 "Certified-Fresh"))  
movies$AStatus = ordered(movies$AStatus)  
  
  
# Remove columns with lots of text (may use them for NLP if time).  
movietext <- subset(movies, select = c(Info, Consensus))  
movies <- subset(movies, select = -c(Info, Consensus))  
  
# Check for and remove NAs  
length(movies[movies=='NA'])

## [1] 2375

movies <- na.omit(movies)  
length(movies[movies=='NA'])

## [1] 0

# Removes Levels with less than 100 values  
movies <- movies[movies$Content != "NC17",]  
movies <- movies[movies$Genre != "Cult Movies",]  
movies <- movies[movies$Genre != "Kids & Family",]  
movies <- movies[movies$Genre != "Musical & Performing Arts",]  
movies <- movies[movies$Genre != "Romance",]  
movies <- movies[movies$Genre != "Science Fiction & Fantasy",]  
movies <- movies[movies$Genre != "Special Interest",]  
movies <- movies[movies$Genre != "Western",]  
  
movies$Content=droplevels(movies$Content)  
movies$Genre=droplevels(movies$Genre)  
  
# Convert Release Date and Stream Date to Date Format  
movies$Release <- as.Date(movies$Release , format = "%Y-%m-%d")  
movies$Stream <- as.Date(movies$Stream , format = "%Y-%m-%d")  
  
# Discretize Release and Stream dates for EDA and analysis.  
Ryear <- as.numeric(format(movies$Release, "%Y"))  
Rdecade <- c()  
for (x in 1:length(Ryear)) {  
 Rdecade[x] <- Ryear[x] - Ryear[x] %% 10  
}  
Syear <- as.numeric(format(movies$Stream, "%Y"))  
  
abbmonth <- ordered(c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",   
 "Sep","Oct", "Nov", "Dec"))  
  
abbmonth <- ordered(abbmonth, levels = c("Jan", "Feb", "Mar", "Apr", "May",   
 "Jun", "Jul", "Aug", "Sep","Oct",   
 "Nov", "Dec") )  
Rmonth <- as.numeric(format(movies$Release, "%m"))  
Rmonth <- abbmonth[Rmonth]  
Smonth <- as.numeric(format(movies$Stream, "%m"))  
Smonth <- abbmonth[Smonth]  
  
Rday <- as.numeric(format(movies$Release, "%d"))  
Rdaybin <- ifelse(Rday <= 31 & Rday > 20, "Late Month",   
 ifelse(Rday <= 20 & Rday > 10, "Mid-Month", "Early Month"))  
Rdaybin <- ordered(Rdaybin, levels = c("Early Month", "Mid-Month",   
 "Late Month"))  
Sday <- as.numeric(format(movies$Stream, "%d"))  
Sdaybin <- ifelse(Sday <= 31 & Sday > 20, "Late Month",   
 ifelse(Sday <= 20 & Sday > 10, "Mid-Month", "Early Month"))  
Sdaybin <- ordered(Sdaybin, levels = c("Early Month", "Mid-Month",   
 "Late Month"))  
dates <- data.frame(Rdecade, Syear, Rmonth, Smonth, Rdaybin, Sdaybin)  
  
movies <- cbind(movies, dates)  
  
# Discretize Runtime for EDA and analysis.  
Runtimebin <- ifelse(movies$Runtime > 240, "240+", ifelse(movies$Runtime <= 240   
 & movies$Runtime > 180, "181-240",   
 ifelse(movies$Runtime <= 180   
 & movies$Runtime > 120, "121-180",  
 ifelse(movies$Runtime <= 120  
 & movies$Runtime > 60,   
 "61-120", "1-60"))))  
Runtimebin <- ordered(Runtimebin, levels = c("1-60", "61-120", "121-180",  
 "181-240", "240+"))  
movies <- cbind(movies, Runtimebin)  
  
# EDA ######################################################  
  
# Status  
gcstatus <- ggplot(movies, aes(x=CStatus)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Critic Status")   
gcstatus <- gcstatus + xlab("Status") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
gastatus <- ggplot(movies, aes(x=AStatus)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Audience Status")   
gastatus <- gastatus + xlab("Status") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
gstatus <- grid.arrange(gcstatus, gastatus, nrow=1)

Chart, bar chart

Description automatically generated

gstatus

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Content Rating  
ccontent <- ggplot(movies, aes(x=Content)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("MPA Rating (Crtitc)")   
ccontent <- ccontent + xlab("Rating") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
acontent <- ggplot(movies, aes(x=Content)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("MPA Rating (Audience)")   
acontent <- acontent + xlab("Rating") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
gcontent <- grid.arrange(ccontent, acontent, nrow=1)

Chart, bar chart

Description automatically generated

gcontent

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Genre  
cgenre <- ggplot(movies, aes(x=Genre)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Genre (Critic)")   
cgenre <- cgenre + xlab("Genre") + ylab("Frequency") +   
 labs(fill = "Critic Status") + theme(axis.text.x = element\_text(angle = 90))   
cgenre <- cgenre + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
agenre <- ggplot(movies, aes(x=Genre)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Genre (Audience)")   
agenre <- agenre + xlab("Genre") + ylab("Frequency") +   
 labs(fill = "Audience Status") + theme(axis.text.x = element\_text(angle = 90))  
agenre <- agenre + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
ggenre <- grid.arrange(cgenre, agenre, nrow=1)

Chart, bar chart, waterfall chart

Description automatically generated

ggenre

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Release Decade   
cdecade <- ggplot(movies, aes(x=Rdecade)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Decade Released")   
cdecade <- cdecade + xlab("Decade") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
adecade <- ggplot(movies, aes(x=Rdecade)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Decade Released")   
adecade <- adecade + xlab("Decade") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
gdecade <- grid.arrange(cdecade, adecade, nrow=1)

Chart, bar chart, waterfall chart

Description automatically generated

gdecade

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Stream Year  
cyear <- ggplot(movies, aes(x=Syear)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Stream Year (Critic)")   
cyear <- cyear + xlab("Year") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
ayear <- ggplot(movies, aes(x=Syear)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Stream Year (Audience)")   
ayear <- ayear + xlab("Year") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
gyear <- grid.arrange(cyear, ayear, nrow=1)

Chart, bar chart

Description automatically generated

gyear

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Release Month  
crmonth <- ggplot(movies, aes(x=Rmonth)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Month Released (Critic)")   
crmonth <- crmonth + xlab("Month") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
armonth <- ggplot(movies, aes(x=Rmonth)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Month Released (Audience)")   
armonth <- armonth + xlab("Month") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
grmonth <- grid.arrange(crmonth, armonth, nrow=1)

Chart, bar chart

Description automatically generated

grmonth

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Stream Month  
csmonth <- ggplot(movies, aes(x=Smonth)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Month Streamed (Critic)")   
csmonth <- csmonth + xlab("Month") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
asmonth <- ggplot(movies, aes(x=Smonth)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Month Streamed (Audience)")   
asmonth <- asmonth + xlab("Month") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
gsmonth <- grid.arrange(csmonth, asmonth, nrow=1)

Chart, bar chart, waterfall chart

Description automatically generated

gsmonth

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Release Day  
crday <- ggplot(movies, aes(x=Rdaybin)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Time of Month Released (Critic)")   
crday <- crday + xlab("Time of Month") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
arday <- ggplot(movies, aes(x=Rdaybin)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Time of Month Released (Audience)")   
arday <- arday + xlab("Time of Month") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
grday <- grid.arrange(crday, arday, nrow=1)

Chart, bar chart

Description automatically generated

grday

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Stream Day  
csday <- ggplot(movies, aes(x=Sdaybin)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Time of Month Streamed (Critic)")   
csday <- csday + xlab("Time of Month") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
asday <- ggplot(movies, aes(x=Sdaybin)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Time of Month Streamed (Audience)")   
asday <- asday + xlab("Time of Month") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
gsday <- grid.arrange(csday, asday, nrow=1)

Chart, bar chart

Description automatically generated

gsday

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Runtime  
crtime <- ggplot(movies, aes(x=Runtimebin)) +   
 geom\_bar(aes(fill = CStatus)) + ggtitle("Runtime (Critic)")   
crtime <- crtime + xlab("Runtime (mins)") + ylab("Frequency") +   
 labs(fill = "Critic Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
artime <- ggplot(movies, aes(x=Runtimebin)) +   
 geom\_bar(aes(fill = AStatus)) + ggtitle("Runtime (Audience)")   
artime <- artime + xlab("Runtime (mins)") + ylab("Frequency") +   
 labs(fill = "Audience Status") + scale\_fill\_manual(values = wes\_palette(n=3,   
 name="Darjeeling1"))  
grtime <- grid.arrange(crtime, artime, nrow=1)

Chart, bar chart, waterfall chart

Description automatically generated

grtime

## TableGrob (1 x 2) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (1-1,2-2) arrange gtable[layout]

# Association Rules ########################################  
  
# Critics  
crules <- subset(movies, select = -c(Release, Stream, AStatus, Release, Stream,   
 Runtime))  
  
crules$Rdecade = as.factor(crules$Rdecade)  
crules$Syear = as.factor(crules$Syear)  
  
# I tweaked the numbers till I got between 20 and 30 rules   
Freshrules <- apriori(data = crules, parameter=list(supp=0.03,conf=0.6),  
 appearance = list(default="lhs", rhs="CStatus=Fresh"),   
 control=list(verbose=F))  
summary(Freshrules)

## set of 23 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 2 3 4   
## 3 11 9   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 3.000 3.000 3.261 4.000 4.000   
##   
## summary of quality measures:  
## support confidence coverage lift   
## Min. :0.03149 Min. :0.6030 Min. :0.05179 Min. :1.636   
## 1st Qu.:0.04259 1st Qu.:0.6089 1st Qu.:0.06311 1st Qu.:1.652   
## Median :0.04958 Median :0.6153 Median :0.08176 Median :1.669   
## Mean :0.05827 Mean :0.6347 Mean :0.09291 Mean :1.722   
## 3rd Qu.:0.05871 3rd Qu.:0.6326 3rd Qu.:0.09479 3rd Qu.:1.716   
## Max. :0.16840 Max. :0.7455 Max. :0.27488 Max. :2.023   
## count   
## Min. : 498.0   
## 1st Qu.: 673.5   
## Median : 784.0   
## Mean : 921.4   
## 3rd Qu.: 928.5   
## Max. :2663.0   
##   
## mining info:  
## data ntransactions support confidence  
## crules 15814 0.03 0.6

Freshrules <- sort(Freshrules, decreasing = TRUE, by="lift")  
inspect(Freshrules[1:23])

## lhs rhs support confidence coverage lift count  
## [1] {Genre=Classics,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.03926900 0.7454982 0.05267485 2.022527 621  
## [2] {Content=NR,   
## Genre=Documentary} => {CStatus=Fresh} 0.04211458 0.7207792 0.05842924 1.955465 666  
## [3] {Content=NR,   
## Genre=Documentary,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.03882636 0.7198124 0.05393955 1.952842 614  
## [4] {Genre=Classics} => {CStatus=Fresh} 0.04799545 0.7080224 0.06778804 1.920855 759  
## [5] {Content=NR,   
## Rdaybin=Early Month} => {CStatus=Fresh} 0.06728216 0.6394231 0.10522322 1.734746 1064  
## [6] {Content=NR,   
## Rdaybin=Early Month,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.05868218 0.6343131 0.09251296 1.720883 928  
## [7] {Content=NR,   
## Sdaybin=Mid-Month,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.04527634 0.6308370 0.07177185 1.711452 716  
## [8] {Content=NR,   
## Sdaybin=Mid-Month} => {CStatus=Fresh} 0.05122044 0.6264501 0.08176299 1.699551 810  
## [9] {Genre=Documentary,   
## Rdecade=2010,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.03237638 0.6236297 0.05191602 1.691899 512  
## [10] {Genre=Documentary,   
## Rdecade=2010} => {CStatus=Fresh} 0.03402049 0.6198157 0.05488807 1.681552 538  
## [11] {Genre=Documentary,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.05166308 0.6156745 0.08391299 1.670317 817  
## [12] {Content=NR,   
## Sdaybin=Early Month} => {CStatus=Fresh} 0.07120273 0.6153005 0.11572025 1.669302 1126  
## [13] {Genre=Documentary} => {CStatus=Fresh} 0.05577337 0.6142061 0.09080562 1.666333 882  
## [14] {Content=NR,   
## Sdaybin=Early Month,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.06241305 0.6138060 0.10168205 1.665247 987  
## [15] {Content=NR} => {CStatus=Fresh} 0.16839509 0.6126064 0.27488302 1.661993 2663  
## [16] {Content=NR,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.14702163 0.6118421 0.24029341 1.659920 2325  
## [17] {Content=NR,   
## Syear=2017,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.04306311 0.6096688 0.07063362 1.654023 681  
## [18] {Content=NR,   
## Rdaybin=Early Month,   
## Sdaybin=Early Month} => {CStatus=Fresh} 0.03149108 0.6080586 0.05178955 1.649655 498  
## [19] {Content=NR,   
## Syear=2016,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.05159985 0.6075949 0.08492475 1.648397 816  
## [20] {Content=NR,   
## Rdaybin=Late Month,   
## Runtimebin=61-120} => {CStatus=Fresh} 0.04363222 0.6073944 0.07183508 1.647853 690  
## [21] {Content=NR,   
## Syear=2016} => {CStatus=Fresh} 0.05874542 0.6052117 0.09706589 1.641931 929  
## [22] {Content=NR,   
## Rdaybin=Late Month} => {CStatus=Fresh} 0.04957632 0.6044719 0.08201594 1.639924 784  
## [23] {Content=NR,   
## Syear=2017} => {CStatus=Fresh} 0.04850133 0.6029874 0.08043506 1.635897 767

Rottenrules <- apriori(data = crules, parameter=list(supp=0.03,conf=0.6),  
 appearance = list(default="lhs", rhs="CStatus=Rotten"),   
 control=list(verbose=F))  
summary(Rottenrules)

## set of 26 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 2 3 4   
## 1 13 12   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 3.000 3.000 3.423 4.000 4.000   
##   
## summary of quality measures:  
## support confidence coverage lift   
## Min. :0.03023 Min. :0.6023 Min. :0.04781 Min. :1.379   
## 1st Qu.:0.03277 1st Qu.:0.6137 1st Qu.:0.05037 1st Qu.:1.405   
## Median :0.03832 Median :0.6340 Median :0.05840 Median :1.452   
## Mean :0.04059 Mean :0.6383 Mean :0.06374 Mean :1.461   
## 3rd Qu.:0.04260 3rd Qu.:0.6647 3rd Qu.:0.06654 3rd Qu.:1.522   
## Max. :0.09359 Max. :0.7076 Max. :0.14772 Max. :1.620   
## count   
## Min. : 478.0   
## 1st Qu.: 518.2   
## Median : 606.0   
## Mean : 642.0   
## 3rd Qu.: 673.8   
## Max. :1480.0   
##   
## mining info:  
## data ntransactions support confidence  
## crules 15814 0.03 0.6

Rottenrules <- sort(Rottenrules, decreasing = TRUE, by="lift")  
inspect(Rottenrules[1:25])

## lhs rhs support confidence coverage lift count  
## [1] {Content=PG-13,   
## Genre=Comedy,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03933224 0.7076223 0.05558366 1.620145 622  
## [2] {Content=PG-13,   
## Genre=Comedy} => {CStatus=Rotten} 0.04116606 0.6970021 0.05906159 1.595829 651  
## [3] {Genre=Comedy,   
## Rdecade=2000,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.04394840 0.6780488 0.06481599 1.552434 695  
## [4] {Genre=Horror,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03364108 0.6742712 0.04989250 1.543785 532  
## [5] {Content=PG-13,   
## Rdecade=2000,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03718224 0.6720000 0.05533072 1.538585 588  
## [6] {Genre=Horror} => {CStatus=Rotten} 0.03421019 0.6695545 0.05109397 1.532986 541  
## [7] {Genre=Comedy,   
## Rdecade=2000} => {CStatus=Rotten} 0.04546604 0.6694600 0.06791451 1.532770 719  
## [8] {Content=PG-13,   
## Rdaybin=Late Month,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03281902 0.6503759 0.05046162 1.489076 519  
## [9] {Content=PG-13,   
## Sdaybin=Early Month,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.04091311 0.6412289 0.06380422 1.468133 647  
## [10] {Content=PG-13,   
## Rdecade=2000} => {CStatus=Rotten} 0.04198811 0.6378482 0.06582775 1.460393 664  
## [11] {Rdecade=1990,   
## Sdaybin=Early Month,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03161755 0.6377551 0.04957632 1.460179 500  
## [12] {Content=PG-13,   
## Genre=Action & Adventure} => {CStatus=Rotten} 0.03035285 0.6349206 0.04780574 1.453690 480  
## [13] {Content=R,   
## Genre=Action & Adventure,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.04281017 0.6344892 0.06747186 1.452702 677  
## [14] {Content=PG-13,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.09358796 0.6335616 0.14771721 1.450578 1480  
## [15] {Content=PG-13,   
## Rdaybin=Mid-Month,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03054256 0.6280884 0.04862780 1.438047 483  
## [16] {Genre=Comedy,   
## Rdaybin=Late Month,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03933224 0.6232465 0.06310864 1.426961 622  
## [17] {Content=PG-13,   
## Rdaybin=Early Month,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03022638 0.6215865 0.04862780 1.423160 478  
## [18] {Genre=Comedy,   
## Sdaybin=Early Month,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.05545719 0.6171710 0.08985709 1.413051 877  
## [19] {Genre=Comedy,   
## Rdaybin=Late Month} => {CStatus=Rotten} 0.04116606 0.6164773 0.06677627 1.411463 651  
## [20] {Content=PG-13,   
## Sdaybin=Early Month} => {CStatus=Rotten} 0.04673074 0.6127695 0.07626154 1.402973 739  
## [21] {Genre=Action & Adventure,   
## Rdecade=2000} => {CStatus=Rotten} 0.03079550 0.6118090 0.05033515 1.400774 487  
## [22] {Genre=Comedy,   
## Sdaybin=Early Month} => {CStatus=Rotten} 0.05710130 0.6093117 0.09371443 1.395057 903  
## [23] {Content=PG-13,   
## Syear=2016,   
## Runtimebin=61-120} => {CStatus=Rotten} 0.03022638 0.6058302 0.04989250 1.387085 478  
## [24] {Content=PG-13,   
## Rdaybin=Late Month} => {CStatus=Rotten} 0.03730871 0.6057495 0.06159100 1.386901 590  
## [25] {Rdecade=1990,   
## Sdaybin=Early Month} => {CStatus=Rotten} 0.03477931 0.6024096 0.05773365 1.379254 550

# Audience  
arules <- subset(movies, select = -c(Release, Stream, CStatus, Release, Stream,   
 Runtime))  
arules$Rdecade = as.factor(arules$Rdecade)  
arules$Syear = as.factor(arules$Syear)  
  
Uprightrules <- apriori(data = arules, parameter=list(supp=0.045,conf=0.6),  
 appearance = list(default="lhs", rhs="AStatus=Upright"),   
 control=list(verbose=F))  
summary(Uprightrules)

## set of 23 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 2 3 4   
## 6 12 5   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 2.500 3.000 2.957 3.000 4.000   
##   
## summary of quality measures:  
## support confidence coverage lift   
## Min. :0.04547 Min. :0.6028 Min. :0.05843 Min. :1.101   
## 1st Qu.:0.05116 1st Qu.:0.6201 1st Qu.:0.08110 1st Qu.:1.133   
## Median :0.05735 Median :0.6364 Median :0.08764 Median :1.163   
## Mean :0.06910 Mean :0.6721 Mean :0.10391 Mean :1.228   
## 3rd Qu.:0.07152 3rd Qu.:0.6819 3rd Qu.:0.10345 3rd Qu.:1.246   
## Max. :0.17516 Max. :0.8329 Max. :0.27488 Max. :1.522   
## count   
## Min. : 719   
## 1st Qu.: 809   
## Median : 907   
## Mean :1093   
## 3rd Qu.:1131   
## Max. :2770   
##   
## mining info:  
## data ntransactions support confidence  
## arules 15814 0.045 0.6

Uprightrules <- sort(Uprightrules, decreasing = TRUE, by="lift")  
inspect(Uprightrules[1:23])

## lhs rhs support confidence coverage lift count  
## [1] {Genre=Documentary} => {AStatus=Upright} 0.07562919 0.8328691 0.09080562 1.521954 1196  
## [2] {Genre=Documentary,   
## Runtimebin=61-120} => {AStatus=Upright} 0.06968509 0.8304446 0.08391299 1.517524 1102  
## [3] {Content=NR,   
## Genre=Documentary} => {AStatus=Upright} 0.04767927 0.8160173 0.05842924 1.491160 754  
## [4] {Genre=Classics} => {AStatus=Upright} 0.05343367 0.7882463 0.06778804 1.440412 845  
## [5] {Runtimebin=121-180} => {AStatus=Upright} 0.09991147 0.7684825 0.13001138 1.404297 1580  
## [6] {Genre=Art House & International} => {AStatus=Upright} 0.08049829 0.6944899 0.11590995 1.269085 1273  
## [7] {Genre=Art House & International,   
## Runtimebin=61-120} => {AStatus=Upright} 0.06361452 0.6693280 0.09504237 1.223105 1006  
## [8] {Content=NR,   
## Rdaybin=Late Month} => {AStatus=Upright} 0.05393955 0.6576715 0.08201594 1.201805 853  
## [9] {Content=NR,   
## Sdaybin=Mid-Month} => {AStatus=Upright} 0.05318073 0.6504254 0.08176299 1.188563 841  
## [10] {Content=NR,   
## Rdaybin=Late Month,   
## Runtimebin=61-120} => {AStatus=Upright} 0.04616163 0.6426056 0.07183508 1.174274 730  
## [11] {Content=NR} => {AStatus=Upright} 0.17516125 0.6372211 0.27488302 1.164434 2770  
## [12] {Content=NR,   
## Rdaybin=Early Month} => {AStatus=Upright} 0.06696598 0.6364183 0.10522322 1.162967 1059  
## [13] {Content=NR,   
## Sdaybin=Early Month} => {AStatus=Upright} 0.07335273 0.6338798 0.11572025 1.158329 1160  
## [14] {Content=NR,   
## Sdaybin=Mid-Month,   
## Runtimebin=61-120} => {AStatus=Upright} 0.04546604 0.6334802 0.07177185 1.157598 719  
## [15] {Content=NR,   
## Sdaybin=Late Month} => {AStatus=Upright} 0.04862780 0.6282680 0.07739977 1.148074 769  
## [16] {Content=NR,   
## Syear=2016} => {AStatus=Upright} 0.06083217 0.6267101 0.09706589 1.145227 962  
## [17] {Content=NR,   
## Runtimebin=61-120} => {AStatus=Upright} 0.14904515 0.6202632 0.24029341 1.133446 2357  
## [18] {Content=NR,   
## Rdaybin=Early Month,   
## Runtimebin=61-120} => {AStatus=Upright} 0.05735424 0.6199590 0.09251296 1.132890 907  
## [19] {Content=NR,   
## Rdaybin=Mid-Month} => {AStatus=Upright} 0.05425572 0.6190476 0.08764386 1.131225 858  
## [20] {Content=NR,   
## Sdaybin=Early Month,   
## Runtimebin=61-120} => {AStatus=Upright} 0.06291893 0.6187811 0.10168205 1.130738 995  
## [21] {Content=NR,   
## Syear=2016,   
## Runtimebin=61-120} => {AStatus=Upright} 0.05242190 0.6172748 0.08492475 1.127985 829  
## [22] {Content=NR,   
## Syear=2017} => {AStatus=Upright} 0.04938662 0.6139937 0.08043506 1.121989 781  
## [23] {Rmonth=Dec} => {AStatus=Upright} 0.04989250 0.6027502 0.08277476 1.101443 789

Spilledrules <- apriori(data = arules, parameter=list(supp=0.025,conf=0.6),  
 appearance = list(default="lhs", rhs="AStatus=Spilled"),   
 control=list(verbose=F))  
summary(Spilledrules)

## set of 26 rules  
##   
## rule length distribution (lhs + rhs):sizes  
## 2 3 4 5   
## 1 5 15 5   
##   
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 2.000 4.000 4.000 3.923 4.000 5.000   
##   
## summary of quality measures:  
## support confidence coverage lift   
## Min. :0.02523 Min. :0.6041 Min. :0.03238 Min. :1.334   
## 1st Qu.:0.02757 1st Qu.:0.6217 1st Qu.:0.04174 1st Qu.:1.373   
## Median :0.03380 Median :0.6580 Median :0.05049 Median :1.453   
## Mean :0.03886 Mean :0.6751 Mean :0.05845 Mean :1.491   
## 3rd Qu.:0.04426 3rd Qu.:0.6896 3rd Qu.:0.06866 3rd Qu.:1.523   
## Max. :0.08366 Max. :0.8320 Max. :0.13083 Max. :1.838   
## count   
## Min. : 399.0   
## 1st Qu.: 436.0   
## Median : 534.5   
## Mean : 614.5   
## 3rd Qu.: 700.0   
## Max. :1323.0   
##   
## mining info:  
## data ntransactions support confidence  
## arules 15814 0.025 0.6

Spilledrules <- sort(Spilledrules, decreasing = TRUE, by="lift")  
inspect(Spilledrules[1:26])

## lhs rhs support confidence coverage lift count  
## [1] {Content=R,   
## Genre=Horror,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.02693816 0.8320312 0.03237638 1.837673 426  
## [2] {Content=R,   
## Genre=Horror} => {AStatus=Spilled} 0.02731757 0.8260038 0.03307196 1.824361 432  
## [3] {Genre=Horror,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.04116606 0.8250951 0.04989250 1.822354 651  
## [4] {Genre=Horror} => {AStatus=Spilled} 0.04173517 0.8168317 0.05109397 1.804103 660  
## [5] {Content=R,   
## Rdecade=2010,   
## Rdaybin=Late Month,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.02554698 0.7026087 0.03636019 1.551823 404  
## [6] {Content=R,   
## Rdecade=2010,   
## Rdaybin=Early Month,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.02832933 0.6945736 0.04078664 1.534076 448  
## [7] {Content=R,   
## Rdecade=2010,   
## Syear=2016,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.03237638 0.6918919 0.04679398 1.528153 512  
## [8] {Content=R,   
## Rdecade=2010,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.07910712 0.6828603 0.11584672 1.508206 1251  
## [9] {Genre=Comedy,   
## Rdecade=2010,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.04622486 0.6694139 0.06905274 1.478507 731  
## [10] {Content=R,   
## Rdecade=2010,   
## Sdaybin=Early Month,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.03326167 0.6675127 0.04982927 1.474308 526  
## [11] {Genre=Action & Adventure,   
## Rdecade=2010,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.03250285 0.6666667 0.04875427 1.472439 514  
## [12] {Content=R,   
## Rdecade=2010,   
## Syear=2016} => {AStatus=Spilled} 0.03402049 0.6633785 0.05128367 1.465177 538  
## [13] {Genre=Comedy,   
## Rdecade=2010} => {AStatus=Spilled} 0.04755280 0.6590710 0.07215126 1.455663 752  
## [14] {Content=R,   
## Rdecade=2010,   
## Rdaybin=Early Month} => {AStatus=Spilled} 0.03003668 0.6569848 0.04571898 1.451056 475  
## [15] {Content=R,   
## Rdecade=2010,   
## Rdaybin=Mid-Month,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.02523081 0.6519608 0.03869989 1.439959 399  
## [16] {Content=R,   
## Rdecade=2010,   
## Rdaybin=Late Month} => {AStatus=Spilled} 0.02662198 0.6506955 0.04091311 1.437165 421  
## [17] {Content=R,   
## Rdecade=2010,   
## Sdaybin=Late Month} => {AStatus=Spilled} 0.02548375 0.6468700 0.03939547 1.428715 403  
## [18] {Content=R,   
## Rdecade=2010} => {AStatus=Spilled} 0.08366005 0.6394393 0.13083344 1.412304 1323  
## [19] {Content=R,   
## Rdecade=2010,   
## Sdaybin=Early Month} => {AStatus=Spilled} 0.03547490 0.6282195 0.05646895 1.387523 561  
## [20] {Content=R,   
## Genre=Action & Adventure,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.04179841 0.6194939 0.06747186 1.368251 661  
## [21] {Genre=Comedy,   
## Syear=2016,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.04508663 0.6146552 0.07335273 1.357564 713  
## [22] {Content=R,   
## Syear=2017,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.03933224 0.6146245 0.06399393 1.357496 622  
## [23] {Content=R,   
## Rdecade=2010,   
## Rdaybin=Mid-Month} => {AStatus=Spilled} 0.02700139 0.6108727 0.04420134 1.349210 427  
## [24] {Genre=Comedy,   
## Sdaybin=Early Month,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.05482484 0.6101337 0.08985709 1.347577 867  
## [25] {Genre=Comedy,   
## Syear=2016} => {AStatus=Spilled} 0.04609839 0.6064892 0.07600860 1.339528 729  
## [26] {Content=PG-13,   
## Genre=Comedy,   
## Runtimebin=61-120} => {AStatus=Spilled} 0.03357784 0.6040956 0.05558366 1.334241 531

# Classification ###########################################  
  
numTotalMovies = nrow(movies)  
trainRatio <- .66  
set.seed(11)  
sample <- sample.int(n = numTotalMovies,   
 size = floor(trainRatio\*numTotalMovies), replace = FALSE)  
train <- movies[sample, ]  
test <- movies[-sample, ]  
length(sample)/nrow(movies)

## [1] 0.6599848

# Removing unnecessary columns for classification  
ctrain <- subset(train, select = -c(Release, Stream, Runtime, AStatus, Syear,  
 Smonth, Sdaybin))  
ctest <- subset(test, select = -c(Release, Stream, Runtime, AStatus, Syear,  
 Smonth, Sdaybin, CStatus))  
ctestanswers <- test$CStatus  
  
# Removing unnecessary columns for classification  
atrain <- subset(train, select = -c(Release, Stream, Runtime, CStatus, Syear,  
 Smonth, Sdaybin))  
atest <- subset(test, select = -c(Release, Stream, Runtime, CStatus, Syear,  
 Smonth, Sdaybin, AStatus))  
atestanswers <- test$AStatus  
  
# Creating a control with cross validation of 5  
control <- trainControl(method ='cv',number = 5)  
  
# Metric for comparison will be accuracy for this project  
metric <- "Accuracy"  
  
set.seed(10)  
  
# Decision Tree  
tree.model <- train(CStatus ~ ., data = ctrain, method="rpart", metric=metric,   
 trControl=control,  
 tuneLength = 5)  
  
# Accuracy of Decision Tree Model  
print(tree.model)

## CART   
##   
## 10437 samples  
## 6 predictor  
## 3 classes: 'Rotten', 'Fresh', 'Certified-Fresh'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 8351, 8350, 8348, 8349, 8350   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.004389667 0.5779415 0.31299833  
## 0.006246834 0.5756421 0.30135015  
## 0.019247003 0.5718117 0.27777287  
## 0.050650008 0.5484346 0.22928754  
## 0.185379031 0.4728546 0.08039388  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.004389667.

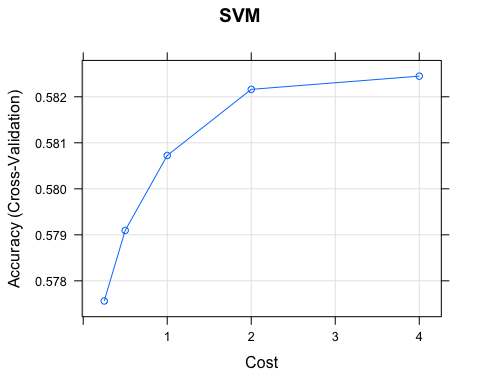
tree\_acc = 58  
Accuracy <- data.frame(tree\_acc)  
  
# Plotting decision tree model  
plot(tree.model, main = "Decision Tree")



# Support Vector Machine (SVM)  
svm.model <- train(CStatus ~ ., data = ctrain, method="svmRadial",metric=metric,  
 trControl=control, tuneLength = 5)  
  
# Accuracy of Support Vector Machine (SVM)  
print(svm.model)

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 10437 samples  
## 6 predictor  
## 3 classes: 'Rotten', 'Fresh', 'Certified-Fresh'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 8349, 8350, 8350, 8349, 8350   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 0.25 0.5775624 0.2982948  
## 0.50 0.5790956 0.3020542  
## 1.00 0.5807243 0.3068085  
## 2.00 0.5821614 0.3107206  
## 4.00 0.5824482 0.3127549  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.02022432  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.02022432 and C = 4.

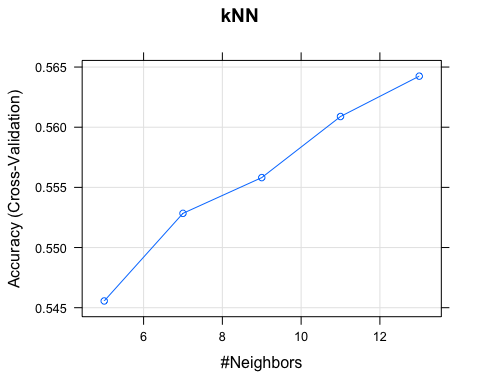
# Plotting Support Vector Machine Model  
plot(svm.model, main = "SVM")



svm\_acc = 58  
Accuracy <- data.frame(cbind(Accuracy,svm\_acc))  
  
# kNN  
knn.model <- train(CStatus ~ ., data = ctrain, method="knn", metric=metric,   
 trControl=control, tuneLength = 5)  
# Accuracy of kNN model  
print(knn.model)

## k-Nearest Neighbors   
##   
## 10437 samples  
## 6 predictor  
## 3 classes: 'Rotten', 'Fresh', 'Certified-Fresh'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 8349, 8350, 8349, 8350, 8350   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 5 0.5455593 0.2603835  
## 7 0.5528421 0.2671223  
## 9 0.5558114 0.2691901  
## 11 0.5608888 0.2745263  
## 13 0.5642428 0.2778768  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 13.

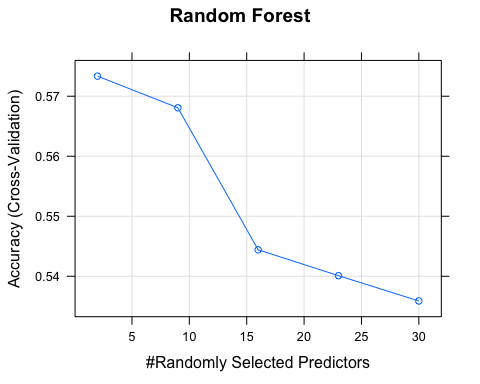
kNN\_acc = 56  
Accuracy <- data.frame(cbind(Accuracy,kNN\_acc))  
  
# Plotting kNN model  
plot(knn.model, main = "kNN")



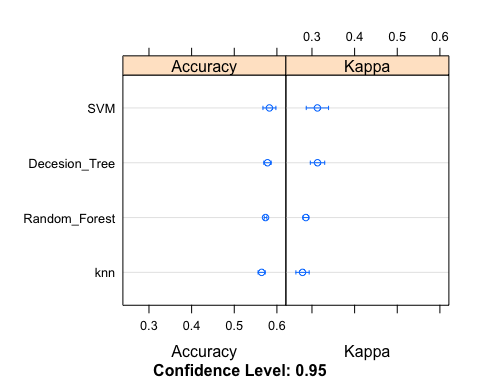
# Random Forest  
# Default number of trees is 500  
set.seed(10)  
rf.model <- train(CStatus ~ ., data = ctrain, method="rf", metric=metric,  
 trControl=control, tuneLength = 5)  
  
# Accuracy of Random Forest Model  
print(rf.model)

## Random Forest   
##   
## 10437 samples  
## 6 predictor  
## 3 classes: 'Rotten', 'Fresh', 'Certified-Fresh'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 8351, 8350, 8348, 8349, 8350   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.5733449 0.2856677  
## 9 0.5680741 0.2940167  
## 16 0.5444092 0.2632030  
## 23 0.5400967 0.2555633  
## 30 0.5358823 0.2491377  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

rf\_acc = 57  
Accuracy <- data.frame(cbind(Accuracy,rf\_acc))  
  
# Plotting random forest model  
plot(rf.model, main = "Random Forest")



# summarize accuracy of models  
results <- resamples(list(Decesion\_Tree=tree.model, knn=knn.model,  
 SVM=svm.model,Random\_Forest=rf.model))  
  
dotplot(results)



# Prediction   
  
# Prediction on the test data using decision tree  
dt <- predict(tree.model, ctest)  
  
prediction <- data.frame(dt)  
  
# Prediction on the test data using svm  
svm <- predict(svm.model, ctest)  
  
prediction <- data.frame(cbind(prediction, svm))  
  
# Prediction on the test data using knn  
knn <- predict(knn.model, ctest)  
  
prediction <- data.frame(cbind(prediction, knn))  
  
# Prediction on the test data using random forest  
random\_f <- predict(rf.model, ctest)  
  
#prediction <- data.frame(random\_f)  
prediction <- data.frame(cbind(prediction, random\_f))  
  
Accuracy

## tree\_acc svm\_acc kNN\_acc rf\_acc  
## 1 58 58 57 57

# Critic Confusion Matricies  
confusionMatrix(prediction$dt, ctestanswers)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Rotten Fresh Certified-Fresh  
## Rotten 1744 620 514  
## Fresh 452 1144 283  
## Certified-Fresh 197 179 244  
##   
## Overall Statistics  
##   
## Accuracy : 0.5825   
## 95% CI : (0.5692, 0.5957)  
## No Information Rate : 0.445   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3191   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: Rotten Class: Fresh Class: Certified-Fresh  
## Sensitivity 0.7288 0.5888 0.23439  
## Specificity 0.6200 0.7860 0.91328  
## Pos Pred Value 0.6060 0.6088 0.39355  
## Neg Pred Value 0.7403 0.7716 0.83246  
## Prevalence 0.4450 0.3614 0.19360  
## Detection Rate 0.3243 0.2128 0.04538  
## Detection Prevalence 0.5352 0.3495 0.11531  
## Balanced Accuracy 0.6744 0.6874 0.57384

confusionMatrix(prediction$svm, ctestanswers)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Rotten Fresh Certified-Fresh  
## Rotten 1813 670 528  
## Fresh 454 1163 333  
## Certified-Fresh 126 110 180  
##   
## Overall Statistics  
##   
## Accuracy : 0.5869   
## 95% CI : (0.5736, 0.6001)  
## No Information Rate : 0.445   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.317   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: Rotten Class: Fresh Class: Certified-Fresh  
## Sensitivity 0.7576 0.5986 0.17291  
## Specificity 0.5985 0.7708 0.94557  
## Pos Pred Value 0.6021 0.5964 0.43269  
## Neg Pred Value 0.7549 0.7724 0.82645  
## Prevalence 0.4450 0.3614 0.19360  
## Detection Rate 0.3372 0.2163 0.03348  
## Detection Prevalence 0.5600 0.3627 0.07737  
## Balanced Accuracy 0.6781 0.6847 0.55924

confusionMatrix(prediction$knn, ctestanswers)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Rotten Fresh Certified-Fresh  
## Rotten 1780 695 576  
## Fresh 477 1178 312  
## Certified-Fresh 136 70 153  
##   
## Overall Statistics  
##   
## Accuracy : 0.5786   
## 95% CI : (0.5652, 0.5918)  
## No Information Rate : 0.445   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3004   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: Rotten Class: Fresh Class: Certified-Fresh  
## Sensitivity 0.7438 0.6063 0.14697  
## Specificity 0.5741 0.7702 0.95249  
## Pos Pred Value 0.5834 0.5989 0.42618  
## Neg Pred Value 0.7365 0.7757 0.82304  
## Prevalence 0.4450 0.3614 0.19360  
## Detection Rate 0.3310 0.2191 0.02845  
## Detection Prevalence 0.5674 0.3658 0.06677  
## Balanced Accuracy 0.6589 0.6883 0.54973

confusionMatrix(prediction$random\_f, ctestanswers)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Rotten Fresh Certified-Fresh  
## Rotten 1918 802 656  
## Fresh 396 1093 289  
## Certified-Fresh 79 48 96  
##   
## Overall Statistics  
##   
## Accuracy : 0.5778   
## 95% CI : (0.5645, 0.5911)  
## No Information Rate : 0.445   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.2881   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: Rotten Class: Fresh Class: Certified-Fresh  
## Sensitivity 0.8015 0.5625 0.09222  
## Specificity 0.5114 0.8005 0.97071  
## Pos Pred Value 0.5681 0.6147 0.43049  
## Neg Pred Value 0.7626 0.7638 0.81665  
## Prevalence 0.4450 0.3614 0.19360  
## Detection Rate 0.3567 0.2033 0.01785  
## Detection Prevalence 0.6279 0.3307 0.04147  
## Balanced Accuracy 0.6564 0.6815 0.53146

# Audience   
  
# Decision Tree  
tree.model <- train(AStatus ~ ., data = atrain, method="rpart", metric=metric,   
 trControl=control,  
 tuneLength = 5)  
  
# Accuracy of Decision Tree Model  
print(tree.model)

## CART   
##   
## 10437 samples  
## 6 predictor  
## 2 classes: 'Spilled', 'Upright'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 8349, 8350, 8349, 8350, 8350   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.003594080 0.6661893 0.3352880  
## 0.004862579 0.6664766 0.3348851  
## 0.005426357 0.6664766 0.3348851  
## 0.052642706 0.6425194 0.3028875  
## 0.066525722 0.5818705 0.1161808  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.005426357.

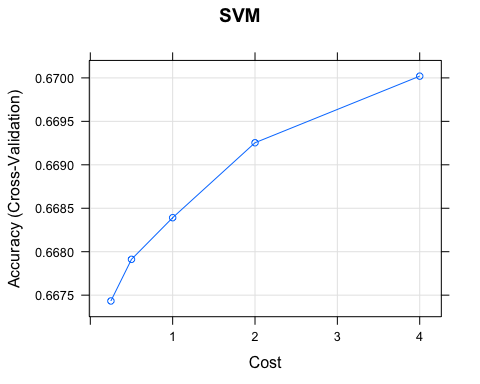
tree\_acc = 67  
Accuracy <- data.frame(tree\_acc)  
  
# Plotting decision tree model  
plot(tree.model, main = "Decision Tree")



# Support Vector Machine (SVM)  
svm.model <- train(AStatus ~ ., data = atrain, method="svmRadial",metric=metric,  
 trControl=control, tuneLength = 5)  
  
# Accuracy of Support Vector Machine (SVM)  
print(svm.model)

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 10437 samples  
## 6 predictor  
## 2 classes: 'Spilled', 'Upright'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 8350, 8350, 8349, 8350, 8349   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 0.25 0.6674331 0.3436484  
## 0.50 0.6679122 0.3425842  
## 1.00 0.6683914 0.3408802  
## 2.00 0.6692538 0.3418020  
## 4.00 0.6700200 0.3411507  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.01969781  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.01969781 and C = 4.

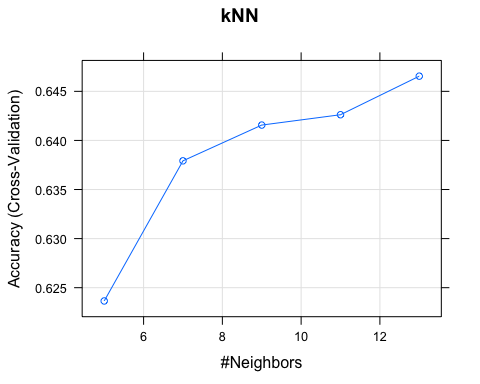
# Plotting Support Vector Machine Model  
plot(svm.model, main = "SVM")



svm\_acc = 67  
Accuracy <- data.frame(cbind(Accuracy,svm\_acc))  
  
# kNN  
knn.model <- train(AStatus ~ ., data = atrain, method="knn", metric=metric,   
 trControl=control, tuneLength = 5)  
  
# Accuracy of kNN model  
print(knn.model)

## k-Nearest Neighbors   
##   
## 10437 samples  
## 6 predictor  
## 2 classes: 'Spilled', 'Upright'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 8350, 8349, 8349, 8350, 8350   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 5 0.6236471 0.2395656  
## 7 0.6379232 0.2686894  
## 9 0.6415635 0.2766390  
## 11 0.6426182 0.2797915  
## 13 0.6465463 0.2883285  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 13.

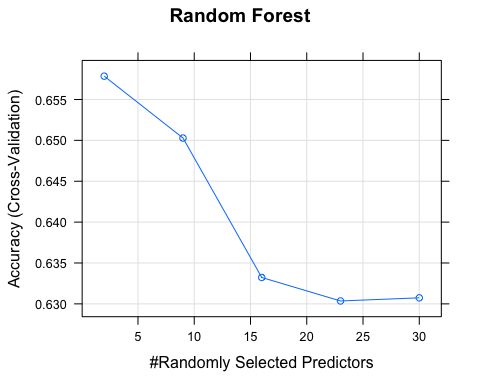
kNN\_acc = 65  
Accuracy <- data.frame(cbind(Accuracy,kNN\_acc))  
  
# Plotting kNN model  
plot(knn.model, main = "kNN")



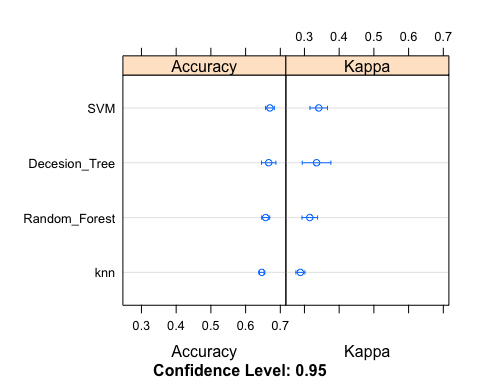
# Random Forest  
# Default number of trees is 500  
set.seed(10)  
rf.model <- train(AStatus ~ ., data = atrain, method="rf", metric=metric,  
 trControl=control, tuneLength = 5)  
  
# Accuracy of Random Forest Model  
print(rf.model)

## Random Forest   
##   
## 10437 samples  
## 6 predictor  
## 2 classes: 'Spilled', 'Upright'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 8349, 8350, 8349, 8350, 8350   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.6578520 0.3154016  
## 9 0.6502829 0.2970492  
## 16 0.6332282 0.2619016  
## 23 0.6303543 0.2570943  
## 30 0.6307374 0.2588718  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

rf\_acc = 66  
Accuracy <- data.frame(cbind(Accuracy,rf\_acc))  
  
# Plotting random forest model  
plot(rf.model, main = "Random Forest")



# summarize accuracy of models  
results <- resamples(list(Decesion\_Tree=tree.model, knn=knn.model,  
 SVM=svm.model,Random\_Forest=rf.model))  
  
dotplot(results)



# Prediction on the test data using decision tree  
dt <- predict(tree.model, atest)  
  
prediction <- data.frame(dt)  
  
# Prediction on the test data using svm  
svm <- predict(svm.model, atest)  
  
prediction <- data.frame(cbind(prediction, svm))  
  
# Prediction on the test data using knn  
knn <- predict(knn.model, atest)  
  
prediction <- data.frame(cbind(prediction, knn))  
  
# Prediction on the test data using random forest  
random\_f <- predict(rf.model, atest)  
  
# prediction <- data.frame(random\_f)  
prediction <- data.frame(cbind(prediction, random\_f))  
  
# Audience Confusion Matricies  
confusionMatrix(prediction$dt, atestanswers)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Spilled Upright  
## Spilled 1856 1292  
## Upright 574 1655  
##   
## Accuracy : 0.653   
## 95% CI : (0.6401, 0.6657)  
## No Information Rate : 0.5481   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3172   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7638   
## Specificity : 0.5616   
## Pos Pred Value : 0.5896   
## Neg Pred Value : 0.7425   
## Prevalence : 0.4519   
## Detection Rate : 0.3452   
## Detection Prevalence : 0.5855   
## Balanced Accuracy : 0.6627   
##   
## 'Positive' Class : Spilled   
##

confusionMatrix(prediction$svm, atestanswers)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Spilled Upright  
## Spilled 1719 1078  
## Upright 711 1869  
##   
## Accuracy : 0.6673   
## 95% CI : (0.6545, 0.6799)  
## No Information Rate : 0.5481   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3371   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7074   
## Specificity : 0.6342   
## Pos Pred Value : 0.6146   
## Neg Pred Value : 0.7244   
## Prevalence : 0.4519   
## Detection Rate : 0.3197   
## Detection Prevalence : 0.5202   
## Balanced Accuracy : 0.6708   
##   
## 'Positive' Class : Spilled   
##

confusionMatrix(prediction$knn, atestanswers)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Spilled Upright  
## Spilled 1488 919  
## Upright 942 2028  
##   
## Accuracy : 0.6539   
## 95% CI : (0.641, 0.6666)  
## No Information Rate : 0.5481   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.3008   
##   
## Mcnemar's Test P-Value : 0.6101   
##   
## Sensitivity : 0.6123   
## Specificity : 0.6882   
## Pos Pred Value : 0.6182   
## Neg Pred Value : 0.6828   
## Prevalence : 0.4519   
## Detection Rate : 0.2767   
## Detection Prevalence : 0.4476   
## Balanced Accuracy : 0.6503   
##   
## 'Positive' Class : Spilled   
##

confusionMatrix(prediction$random\_f, atestanswers)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Spilled Upright  
## Spilled 1710 1106  
## Upright 720 1841  
##   
## Accuracy : 0.6604   
## 95% CI : (0.6476, 0.6731)  
## No Information Rate : 0.5481   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3239   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.7037   
## Specificity : 0.6247   
## Pos Pred Value : 0.6072   
## Neg Pred Value : 0.7189   
## Prevalence : 0.4519   
## Detection Rate : 0.3180   
## Detection Prevalence : 0.5237   
## Balanced Accuracy : 0.6642   
##   
## 'Positive' Class : Spilled   
##

# Plot for results section  
rulesplot <- ggplot(movies, aes(x=Genre)) +   
 geom\_bar(aes(fill = Content)) + ggtitle("Content Rating by Genre")   
rulesplot <- rulesplot + xlab("Genre") + ylab("Frequency") +   
 labs(fill = "Content Rating") + theme(axis.text.x = element\_text(angle = 90))   
rulesplot <- rulesplot + scale\_fill\_manual(values = wes\_palette(n=5,   
 name="Darjeeling1"))  
rulesplot

