What makes a Movie have High Audience Ratings and High Profit? What is the Relationship between a Movie’s High Audience Ratings and High Profit?

Marie Hasegawa - [mhasegawa7045@floridapoly.edu](mailto:mhasegawa7045@floridapoly.edu)

# Appendix

* Appendix
* Library of Packages
* Project Overview
* Exploratory Data Analysis
  + Profitability Column
  + AudienceScore Column
* Methods
  + Profitability vs. AudienceScore Grouped by HighProfit
  + Profitability and AudienceScore K-Means Clustering
    - Elbow Method and Silhouette method to determine optimal clusters
    - K-Means Clustering of k=2,3,4
  + Profitability Column
    - Decision Tree Profitability
      * With Categorical Variables
      * Removed Categorical Variables
    - Bagging
    - Random Forest
    - Boosting
    - Model Comparison High Profitability
  + AudienceScore Column
    - Decision Tree Profitability
      * With Categorical Variables
      * Removed Categorical Variables
    - Bagging
    - Random Forest
    - Boosting
    - Model Comparison High AudienceScore
* Conclusion

# Library of Packages

| **Variables** | **Description** |
| --- | --- |
| Movie | Title of movie |
| LeadStudio | Studio that released the movie |
| RottenTomatoes | Rotten Tomatoes rating (reviewers) |
| AudienceScore | Audience rating (via Rotten Tomatoes) |
| Story | General theme - one of 21 themes |
| Genre | Type of Movie: Action, Adventure, Animation, Comedy, Drama, Fantasy, Horror, Romance, or Thriller |
| TheatersOpenWeek | Number of screens for opening weekend |
| BOAverageOpenWeek | Average box office income per theater - opening weekend |
| DomesticGross | Gross income for domestic viewers (in millions) |
| ForeignGross | Gross income for foreign viewers (in millions) |
| WorldGross | Gross income for all viewers (in millions) |
| Budget | Production budget (in millions) |
| Profitability | WorldGross/Budget |
| OpeningWeekend | Opening weekend gross (in millions) |

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.4.3

## Warning: package 'ggplot2' was built under R version 4.4.3

## Warning: package 'tibble' was built under R version 4.4.3

## Warning: package 'tidyr' was built under R version 4.4.3

## Warning: package 'readr' was built under R version 4.4.3

## Warning: package 'purrr' was built under R version 4.4.3

## Warning: package 'dplyr' was built under R version 4.4.3

## Warning: package 'forcats' was built under R version 4.4.3

## Warning: package 'lubridate' was built under R version 4.4.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.2 ✔ tibble 3.3.0  
## ✔ lubridate 1.9.4 ✔ tidyr 1.3.1  
## ✔ purrr 1.1.0   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ISLR)

## Warning: package 'ISLR' was built under R version 4.4.3

library(caret)

## Warning: package 'caret' was built under R version 4.4.3

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(tidyverse)  
#install.packages("skimr")  
library(skimr)

## Warning: package 'skimr' was built under R version 4.4.3

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.4.3

## Loading required package: rpart

library(pROC)

## Warning: package 'pROC' was built under R version 4.4.3

## Type 'citation("pROC")' for a citation.  
##   
## Attaching package: 'pROC'  
##   
## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(ggplot2)  
#install.packages("factoextra")  
library(factoextra)

## Warning: package 'factoextra' was built under R version 4.4.3

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

#install.packages("gridExtra")  
library(gridExtra)

## Warning: package 'gridExtra' was built under R version 4.4.3

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

movie <- read\_csv("https://raw.githubusercontent.com/reisanar/datasets/master/HollywoodMovies.csv")

## Rows: 970 Columns: 16  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (4): Movie, LeadStudio, Story, Genre  
## dbl (12): RottenTomatoes, AudienceScore, TheatersOpenWeek, OpeningWeekend, B...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Project Overview

The box office industry is one of the biggest global markets in the entertainment industry. According to IMDbPro, the industry has a yearly average box office revenue of around 11 to 14 million dollars and a yearly total gross of around 9 to 12 billion dollars. The project wants to determine what makes a Hollywood movie have a high Profitability and what makes it have a high AudienceScore, and how Profitability and AudienceScore correlate with each other. Movie studios would find this information resourceful since it could help them determine the best kind of movies that will lead to larger profits and more positive feedback on their movies for better publicity.

This will be done with EDA, K-means clustering, decision trees, bagging, boosting, and random forests that will help determine the pattern for high-earning and highly-rated movies. This project will be done in RStudio and RCloud with the dataset, HollywoodMovies, provided by Dr. Reinaldo Sanchez-Arias.

# Explatory Data Analysis

movie<-movie %>%  
 na.omit()  
sample\_n(movie, size=20)

## # A tibble: 20 × 16  
## Movie LeadStudio RottenTomatoes AudienceScore Story Genre TheatersOpenWeek  
## <chr> <chr> <dbl> <dbl> <chr> <chr> <dbl>  
## 1 Wall St… Independe… 54 43 Riva… Drama 3565  
## 2 Public … Universal 67 65 Unde… Biog… 3334  
## 3 Scream 4 Weinstein 58 57 Mons… Horr… 3305  
## 4 Pandorum Independe… 28 49 Mons… Acti… 2506  
## 5 Nancy D… Independe… 49 61 The … Thri… 2612  
## 6 Night a… Fox 43 60 Quest Acti… 4096  
## 7 Pride &… Warner Br… 34 54 Temp… Crime 2585  
## 8 Max Pay… Fox 16 36 Reve… Acti… 3376  
## 9 Apollo … Weinstein 23 31 Mons… Horr… 3328  
## 10 My Sist… Warner Br… 47 73 Esca… Drama 2606  
## 11 Repo Men Universal 22 43 Resc… Acti… 2521  
## 12 Smokin'… Universal 27 64 Purs… Acti… 2218  
## 13 Just Go… Happy Mad… 19 63 Come… Roma… 3548  
## 14 Anonymo… Relativit… 46 66 Trag… Drama 265  
## 15 Ocean's… Warner Br… 70 74 Reve… Thri… 3565  
## 16 Race to… Buena Vis… 42 50 Quest Acti… 3187  
## 17 Star Tr… Paramount 94 91 Reve… Acti… 3849  
## 18 The Tre… Independe… 84 61 Disc… Drama 4  
## 19 Everybo… Miramax 46 55 Disc… Drama 2133  
## 20 Mamma M… Universal 53 76 Love Come… 2976  
## # ℹ 9 more variables: OpeningWeekend <dbl>, BOAvgOpenWeekend <dbl>,  
## # DomesticGross <dbl>, ForeignGross <dbl>, WorldGross <dbl>, Budget <dbl>,  
## # Profitability <dbl>, OpenProfit <dbl>, Year <dbl>

movie %>%  
 filter(Profitability>=400)

## # A tibble: 147 × 16  
## Movie LeadStudio RottenTomatoes AudienceScore Story Genre TheatersOpenWeek  
## <chr> <chr> <dbl> <dbl> <chr> <chr> <dbl>  
## 1 Shrek t… Paramount 42 57 Quest Anim… 4122  
## 2 Transfo… Paramount 57 89 Mons… Acti… 4011  
## 3 Harry P… Warner Br… 78 82 Quest Adve… 4285  
## 4 The Bou… Universal 93 91 Purs… Thri… 3660  
## 5 Alvin a… Fox 26 73 Come… Anim… 3475  
## 6 300 Warner Br… 60 90 Sacr… Acti… 3103  
## 7 Ratatou… Disney 97 84 Tran… Anim… 3940  
## 8 The Sim… Fox 90 78 Matu… Come… 3922  
## 9 Knocked… Universal 91 83 Love Come… 2871  
## 10 Juno Fox 94 89 Matu… Come… 1019  
## # ℹ 137 more rows  
## # ℹ 9 more variables: OpeningWeekend <dbl>, BOAvgOpenWeekend <dbl>,  
## # DomesticGross <dbl>, ForeignGross <dbl>, WorldGross <dbl>, Budget <dbl>,  
## # Profitability <dbl>, OpenProfit <dbl>, Year <dbl>

The movie(HollywoodMovies) dataset originally had many NA values in all columns, which needed to be removed. The code above is used to remove NA values with na.omit() since it needs to be removed in order to train the code and create decision and random trees.

## Profitability Column

summary(movie$Profitability)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 18.17 151.52 253.64 356.04 397.08 6694.40

The variable, Profitability, in the dataset is the division between WorldGross and Budget in millions. The variables is used to determine if the movie made a huge profit beween WorldGross and Budget in millions. The variable, Profitability, will be used in this section to see what factors affect Profitabilityof a movie and how highly-profited movies and lowly profited movies differ from each other. The variable, HighProfit, will be mutated from the variable, Profitability, in order to differ between high-profited movies and low-profited movies. Based on the summarization shown above, the project will define the variable,HighProfit, to determine a movie is highly-profitable if Profitability has a value of 400 or more, since the mean is 356.04 and the third-quarter is 397.08. The HighProfit variable can not be in the thousands like in the max value of 6694.40, since those values are considered an outlier becuase there are too few points to consider it.

raw\_A<-ggplot(data=movie) +  
 geom\_histogram(aes(x=Profitability)) +  
 labs(title="Bar Chart of Profitability")

rawB\_Prof<-ggplot(data=movie) +  
 geom\_histogram(aes(x=Profitability)) +  
 xlim(0,2000) +  
 labs(title="Bar Chart of Profitability Removed Outliers")

movieB<-movie %>%  
 mutate(HighProfit=ifelse(movie$Profitability >= 400, "Yes", "No"))   
  
mBHP\_F<-movieB %>%  
 ggplot(aes(Profitability, colour = HighProfit)) +  
 geom\_freqpoly() +  
 xlim(0,2000) +  
 labs(title="Frequency Plot of HighProfit")  
  
mBHP\_D<-movieB %>%  
 ggplot(aes(Profitability, colour = HighProfit)) +  
 geom\_bar() +  
 geom\_density(aes(y = ..count.., group=HighProfit),color="black") +  
 xlim(0,2000) +  
 labs(title="Bar Chart of HighProfit with Density Analysis")  
  
grid.arrange(raw\_A,rawB\_Prof,mBHP\_F, mBHP\_D)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_bin()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_bar()`).

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

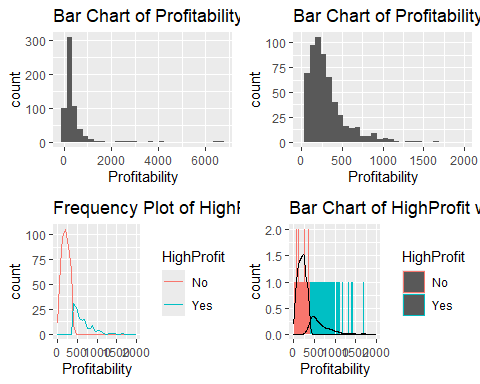
## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_bin()`).

## Warning: Removed 4 rows containing missing values or values outside the scale range  
## (`geom\_path()`).

## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.  
## ℹ Please use `after\_stat(count)` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_count()`).

## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_density()`).



grid.arrange(raw\_A,rawB\_Prof,mBHP\_F, mBHP\_D, ncol=1)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_bin()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_bar()`).

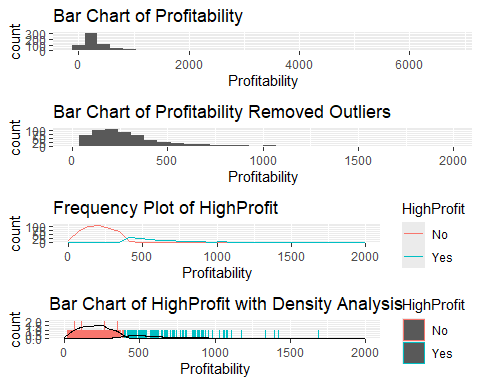
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_bin()`).

## Warning: Removed 4 rows containing missing values or values outside the scale range  
## (`geom\_path()`).

## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_count()`).

## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_density()`).



#aes(group=HighProfit),method="lm",color="black", size=0.3

As you can see in the histogram, Bar Chart of Profitibility, Profitability accumulates mostly in the range of ~0 to ~2000 and has outliers in the range of ~2000 to 6500. Due to this, the outliers will be omitted for the EDA and in the Bar Chart of Profitibility, so it can be easier to analyze the difference between high-profited movies and low-profited movies.

The Frequency Plot of HighProfit and the Bar Chart of HighProfit with Density Analysis shows that low-profited movies occur more frequently compared to high-profited movies. The reason might be that a movie’s success is dependent on the movie’s advertisement, availability globally, and its name/franchise recognition.

For instance, the movie, **Shrek the Third** is an extremely profitable movie of 499.35 but it is a sequel of the other Shrek film and is based on a very popular icon, Shrek, which could contribute a film’s Profitability.

## AudienceScore Column

summary(movie$AudienceScore)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 19.00 48.00 60.00 60.28 73.00 96.00

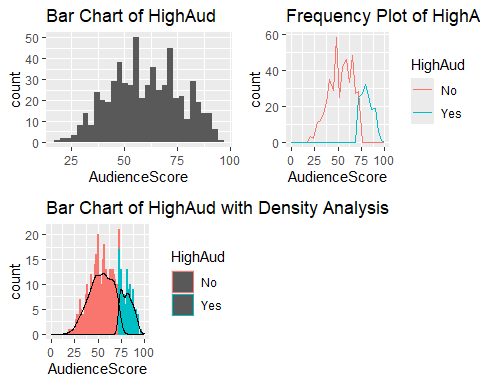
The variable, AudienceScore, in the dataset is a collection of the Rotten Tomatoes ratings from the audience. The variable HighAud is used to determine if the movie made a good impression on the Rotten Tomatoes audience. The variable, AudienceScore, will be used in this project to see what factors affect AudienceScore ratings of a movie and how highly-rated movies based on audience reviews differ from lowly-rated films. The variable, HighAud, will be mutated from the variable, AudienceScore, in order to differentiate between highly-rated by-audience movies and low-rated movies. Based on the summarization shown above, the project will define the variable, HighAud, to determine a movie is highly-rated by critics if AudienceScore has a value of 73 or more since the third-quarter is 73.

raw\_B<-ggplot(data=movie) +  
 geom\_histogram(aes(x=AudienceScore)) +  
 labs(title="Bar Chart of HighAud ")

movieB<-movieB %>%  
 mutate(HighAud=ifelse(movie$AudienceScore >= 73, "Yes", "No"))   
  
Freq\_mB<-movieB %>%  
 ggplot(aes(AudienceScore, colour = HighAud)) +  
 geom\_freqpoly() +  
 xlim(0,100) +   
 labs(title="Frequency Plot of HighAud with Density Analysis")  
  
Bar\_mB<-  
 movieB %>%  
 ggplot(aes(AudienceScore, colour = HighAud)) +  
 geom\_bar() +  
 geom\_density(aes(y = ..count.., group=HighAud),color="black") +  
 xlim(0,100)+  
 labs(title="Bar Chart of HighAud with Density Analysis")  
   
grid.arrange(raw\_B,Freq\_mB, Bar\_mB, nrow = 2)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

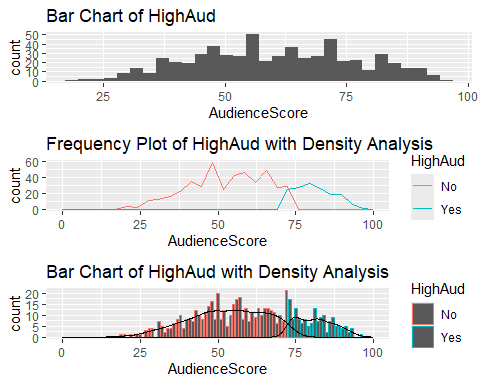
## Warning: Removed 4 rows containing missing values or values outside the scale range  
## (`geom\_path()`).



grid.arrange(raw\_B,Freq\_mB, Bar\_mB)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 4 rows containing missing values or values outside the scale range  
## (`geom\_path()`).



As you can see in the histogram, Bar Chart of HighAud, AudienceScore has a range of 0 to 100 with 100 as a perfect score and 0 as the worst score. There is no significant difference between the frequencies of the highly-rated and lowly-rated movies, as you can see in the Bar Chart of HighAud. Based on the Frequency Chart of HighAud with Density Analysis, there is a ~5-count average difference between the highly-scored and lowly-scored movies.The Frequency Plot of HighAud and the Bar Chart of HighAud with Density Analysis shows that lowly-rated movies occur more frequently compared to highly-rated movies.

# Methods

## Profitability vs. AudienceScore Grouped by HighProfit

The purpose of this section is to see the relationship between Profitability and AudienceScore by clustering the points with HighProfit.

movieB

## # A tibble: 591 × 18  
## Movie LeadStudio RottenTomatoes AudienceScore Story Genre TheatersOpenWeek  
## <chr> <chr> <dbl> <dbl> <chr> <chr> <dbl>  
## 1 Spider-… Sony 61 54 Meta… Acti… 4252  
## 2 Shrek t… Paramount 42 57 Quest Anim… 4122  
## 3 Transfo… Paramount 57 89 Mons… Acti… 4011  
## 4 Pirates… Disney 45 74 Resc… Acti… 4362  
## 5 Harry P… Warner Br… 78 82 Quest Adve… 4285  
## 6 I Am Le… Warner Br… 69 69 Quest Thri… 3606  
## 7 The Bou… Universal 93 91 Purs… Thri… 3660  
## 8 Nationa… Disney 31 72 The … Thri… 3832  
## 9 Alvin a… Fox 26 73 Come… Anim… 3475  
## 10 300 Warner Br… 60 90 Sacr… Acti… 3103  
## # ℹ 581 more rows  
## # ℹ 11 more variables: OpeningWeekend <dbl>, BOAvgOpenWeekend <dbl>,  
## # DomesticGross <dbl>, ForeignGross <dbl>, WorldGross <dbl>, Budget <dbl>,  
## # Profitability <dbl>, OpenProfit <dbl>, Year <dbl>, HighProfit <chr>,  
## # HighAud <chr>

movieBclusHighProf<-movieB %>%  
ggplot(aes(x = AudienceScore, y = Profitability, color = as.factor(HighProfit))) +   
 geom\_point() +  
 geom\_smooth(aes(group=HighProfit),method="lm",color="black", size=0.3)+  
 labs(title = "ScatterPlot of AudienceScore vs Profitability with Linear Method")

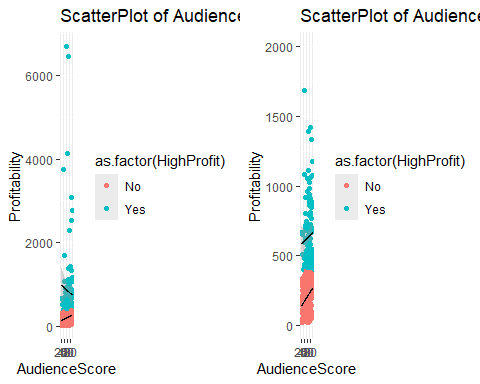
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

movieBclusHighProf\_2000<-movieB %>%  
ggplot(aes(x = AudienceScore, y = Profitability, color = as.factor(HighProfit))) +   
 geom\_point() +  
 geom\_smooth(aes(group=HighProfit),method="lm",color="black", size=0.3)+  
 ylim(0,2000)+  
 labs(title = "ScatterPlot of AudienceScore vs Profitability with Linear Method without Outliers")  
grid.arrange(movieBclusHighProf,movieBclusHighProf\_2000, nrow = 1)

## `geom\_smooth()` using formula = 'y ~ x'  
## `geom\_smooth()` using formula = 'y ~ x'

## Warning: Removed 8 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

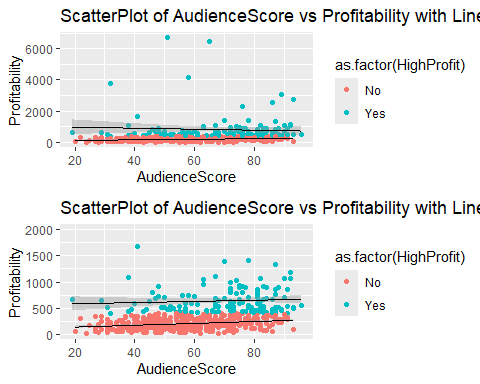
## Warning: Removed 8 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



grid.arrange(movieBclusHighProf,movieBclusHighProf\_2000)

## `geom\_smooth()` using formula = 'y ~ x'  
## `geom\_smooth()` using formula = 'y ~ x'

## Warning: Removed 8 rows containing non-finite outside the scale range (`stat\_smooth()`).  
## Removed 8 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



The ggplots shown above shows the relationship between AudienceScore and Profitability that is clustered with HighProfit.

Based on the ScatterPlot of AudienceScore vs Profitability with Linear Method graph, low-profit movies leads to a slight positive correlation for AudienceScore and Profitability, but high-profit movies has a negative correlation. In other words, as low-profit movies’ AudienceScore increases,it leads to a slight increase in profits. For high-profit movies, as AudienceScore increases, it leads to a decrease in profits.

Based on the ScatterPlot of AudienceScore vs Profitability with Linear Method without Outliers graph, both low-profit and high-profit movies has a positive correlation for AudienceScore and Profitability. In other words, movies, regardless if their high-profit or low-profit, will increase in Profitability as AudienceScore increases.

These scatterplots show that as more highly profitable outliers appear, the slopes of both lowly profitable and highly profitable movies decrease, suggesting a negative correlation between highly profitable outliers and the slopes of both groups.

## Profitability and AudienceScore K-Means Clustering

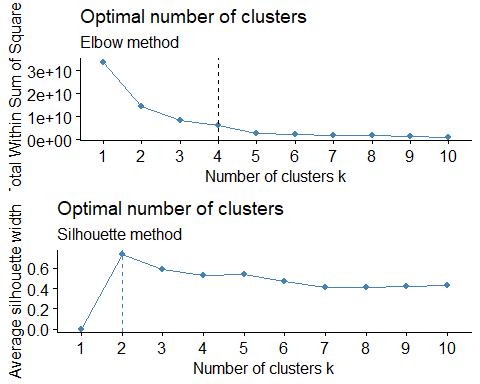
### Elbow Method and Silhouette method to determinine optimal clusters

myvars<-c("RottenTomatoes", "AudienceScore", "TheatersOpenWeek", "DomesticGross", "ForeignGross","WorldGross","Budget","Profitability","OpeningWeekend","BOAvgOpenWeekend")

The code above is used to remove the categorical variables, Movie and LeadStudio, so that the data can be used for k-means clustering, since clustering does not respond well to data with string-based columns.

movieB<-movieB[myvars] %>%  
 na.omit()  
  
movieB<-data.frame(movieB)

set.seed(123)  
# Elbow method  
ELBmovie<-fviz\_nbclust(movieB, kmeans, method = "wss") +  
 geom\_vline(xintercept = 4, linetype = 2)+  
 labs(subtitle = "Elbow method")  
  
# Silhouette method  
SLmovie<-fviz\_nbclust(movieB, kmeans, method = "silhouette")+  
 labs(subtitle = "Silhouette method")  
set.seed(123)  
grid.arrange(ELBmovie, SLmovie, nrow = 2)



In order to find the most optimal number clusters to use for k-means clustering. The elbow method and the silhouette method will be used to determine the best number of clusters. The elbow method computes k-means clustering for every different value of k. Then for every k value has the total within-cluster sum of squares (wss) calculated. Based on this, the best cluster for the elbow method would be 4. The silhouette method measures the quality of a cluster which determines how well each object lies within their cluster. Based on this and the Optimal number of clusters Silhouette Method, the most optimal number of clusters is 2. The project will choose the k values 2, 3, and 4 for K-means clustering since they are the most optimal k-values based on the elbow method and the silhouette method.

### K-Means Clustering of k=2,3,4

myvars<-c("RottenTomatoes", "AudienceScore", "TheatersOpenWeek", "DomesticGross", "ForeignGross","WorldGross","Budget","Profitability","OpeningWeekend","BOAvgOpenWeekend")

The code above is used to remove the categorical variables, Movie and LeadStudio, so the dataset can be used for kmeans, since kmeans does not respond to datasets with columns being a collection of string values.

movieB<-movieB[myvars] %>%  
 na.omit()  
  
movieB<-data.frame(movieB)

movieB\_clust<-movieB[,5]  
mcls2<-kmeans(x=movieB\_clust[!is.na(movieB\_clust)], centers=2)  
mcls3<-kmeans(x=movieB\_clust[!is.na(movieB\_clust)], centers=3)  
mcls4<-kmeans(x=movieB\_clust[!is.na(movieB\_clust)], centers=4)

mcls2

## K-means clustering with 2 clusters of sizes 58, 533  
##   
## Cluster means:  
## [,1]  
## 1 471.00779  
## 2 57.84004  
##   
## Clustering vector:  
## [1] 1 1 1 1 1 1 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2  
## [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [75] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 1 2 2 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2  
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2 2  
## [223] 1 2 1 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [260] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [297] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [334] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2  
## [371] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [408] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2  
## [445] 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [482] 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2  
## [519] 2 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 1 1 2 2 2 2 1 2 2 2 2 2 2  
## [556] 1 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 1 2 1 1 2 2 2 2 2 2 2 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 3761951 1794220  
## (between\_SS / total\_SS = 61.6 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

mcls2 is a k-means clustering with 2 clusters or centers of size 58 and 533. The cluster mean of 1 is 471.00779 and the cluster mean of 2 is 57.84004. The sum of squares in cluster 1 is 3761951 and in cluster 2 is 1794220.

mcls3

## K-means clustering with 3 clusters of sizes 84, 13, 494  
##   
## Cluster means:  
## [,1]  
## 1 298.49229  
## 2 795.62923  
## 3 46.01348  
##   
## Clustering vector:  
## [1] 2 1 1 2 2 1 1 1 3 1 1 1 3 3 3 3 1 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 1 3  
## [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [75] 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 3 1 1 1 3 1 3 3 3 1 3 1 3 3 3 3 3 1  
## [112] 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [186] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 2 1 1 1 3 3 1  
## [223] 1 1 2 1 1 2 3 1 3 3 3 1 1 1 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3  
## [260] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [297] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [334] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3  
## [371] 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 2 3 1 1 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3  
## [408] 3 3 1 3 3 3 3 3 3 1 3 3 3 1 1 1 3 3 1 3 1 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3  
## [445] 3 1 3 1 1 3 3 3 3 3 3 3 3 3 1 3 1 3 3 2 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
## [482] 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3  
## [519] 3 3 3 1 3 3 3 3 1 3 3 3 3 3 3 3 3 3 2 3 1 3 1 3 1 1 3 3 3 3 1 3 3 3 3 3 3  
## [556] 1 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 3 1 3 2 1 3 3 3 3 3 1 3 3  
##   
## Within cluster sum of squares by cluster:  
## [1] 864482.1 1754172.2 828400.7  
## (between\_SS / total\_SS = 76.2 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

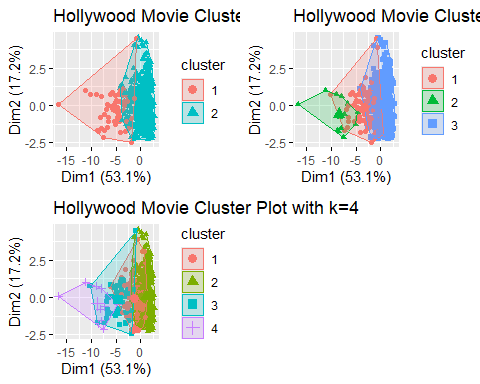
mcls3 is a k-means clustering with 3 clusters or centers of size 84, 13, and 494. The cluster mean of 1 is 298.49229, the cluster mean of 2 is 795.62923, and the cluster mean 3 of 46.01348. The sum of squares in cluster 1 is 864482.1, cluster 2 is 1754172.2, and cluster 3 is 828400.7.

mcls4

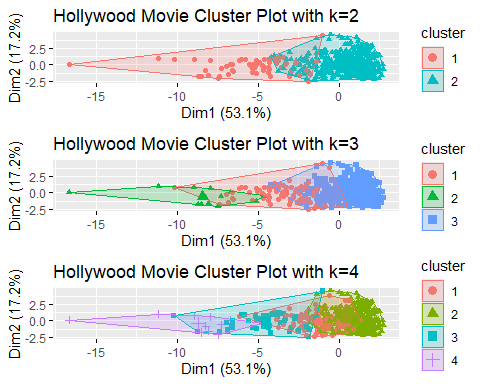
## K-means clustering with 4 clusters of sizes 115, 420, 45, 11  
##   
## Cluster means:  
## [,1]  
## 1 153.82391  
## 2 32.55724  
## 3 391.06960  
## 4 835.02455  
##   
## Clustering vector:  
## [1] 3 3 3 4 4 3 1 1 1 1 3 3 2 2 2 1 1 1 1 1 1 2 2 2 2 1 1 2 2 2 2 2 1 2 1 3 2  
## [38] 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [75] 2 2 2 2 2 2 2 2 2 2 3 1 3 3 3 3 3 1 3 1 1 3 3 1 1 1 2 1 1 1 3 1 2 2 1 1 1  
## [112] 2 1 2 1 2 1 1 2 1 2 2 1 2 2 2 1 2 1 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2  
## [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 3 4 3 3 1 1 2 1  
## [223] 3 1 4 1 1 3 1 1 1 2 2 1 3 1 1 1 1 1 1 2 2 1 1 2 2 1 1 1 2 2 2 2 2 2 1 2 2  
## [260] 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [297] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [334] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 1 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2  
## [371] 3 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 1 1 4 2 3 3 3 2 2 2 2 2 1 2 2 2 2 2 1 2 2  
## [408] 2 2 1 2 2 2 1 2 2 1 1 2 2 1 1 1 2 2 1 2 3 1 2 2 1 2 2 3 1 2 2 2 2 3 2 1 2  
## [445] 2 1 2 3 1 2 2 2 2 2 2 1 1 2 1 2 3 2 2 4 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1  
## [482] 1 1 2 2 1 1 3 2 2 2 2 2 2 2 2 2 2 2 3 1 2 1 2 1 1 2 2 2 4 2 2 1 1 1 2 2 2  
## [519] 1 1 2 3 2 2 2 1 3 2 2 1 2 2 2 2 1 2 4 2 3 1 1 2 3 3 2 2 2 2 3 2 2 2 2 1 2  
## [556] 3 2 2 2 2 2 2 1 2 2 1 3 2 2 2 2 2 2 2 2 3 2 1 2 1 2 4 3 2 2 2 2 2 1 2 2  
##   
## Within cluster sum of squares by cluster:  
## [1] 268727.5 285472.8 292001.1 1641993.1  
## (between\_SS / total\_SS = 82.8 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss" "tot.withinss"  
## [6] "betweenss" "size" "iter" "ifault"

mcls4 is a k-means clustering with 4 clusters or centers of size 115, 420, 45, and 11. The cluster mean of 1 is 153.82391, cluster mean of 2 is 32.55724, cluster mean of 3 is 391.06960, and cluster mean of 4 is 835.02455. The sums of squares in cluster 1, 2, 3, and 4 are 268727.5, 285472.8, 292001.1, and 1641993.1.

mB2<-fviz\_cluster(mcls2, geom = "point", data=movieB)+ggtitle("Hollywood Movie Cluster Plot with k=2")  
mB3<-fviz\_cluster(mcls3, geom = "point", data=movieB)+ggtitle("Hollywood Movie Cluster Plot with k=3")  
mB4<-fviz\_cluster(mcls4, geom = "point", data=movieB)+ggtitle("Hollywood Movie Cluster Plot with k=4")  
  
grid.arrange(mB2, mB3, mB4, nrow = 2)



grid.arrange(mB2, mB3, mB4, ncol = 1)



The Hollywood Movie Cluster Plot with k=2 shows 2 clusters. Cluster 1 takes up the Dim1 (53.1%) range between -17 to -2 and Dim2 (17.2%) range between -2.5 to 6. Cluster 2 takes up the Dim2 (53.1%) range between -17 to -2 and Dim1 (17.2%) range between -2.5 to 6.

The Hollywood Movie Cluster Plot with k=3 shows 3 clusters. Cluster 1 has Dim1 (53.1%) range between -10 to 0 and Dim2 (17.2%) range between 0 to 5. The Cluster 2 has Dim1 (53.1%) range between -0.5 to 0.5 and Dim2 (17.2%) range between -2.5 to 3.75. Cluster 3 has a Dim1 (53.1%) range between -0.5 to 0.5 and Dim2 (17.2%) range between -2.5 to 5.

The Hollywood Movie Cluster Plot with k=4 shows 4 clusters. Cluster 1 has Dim1 (53.1%) range between -6 to 0 and Dim2 (17.2%) range between -2.5 to 5. The Cluster 2 has Dim1 (53.1%) range between -2.5 to 5 and Dim2 (17.2%) range between -2.5 to 4.75. Cluster 3 has a Dim1 (53.1%) range between -10 to 7.5 and Dim2 (17.2%) range between -2.5 to 5. Cluster 4 has a Dim1 (53.1%) range between -6 to 0 and Dim2 (17.2%) range between -2.5 to 5.

As you can see, the clusters in all k-means clustering are overlapped with the other clusters within their graphs. Based on this, Hollywood Movie Cluster Plot with k=3 and Hollywood Movie Cluster Plot with k=2 have better clusters than Hollywood Movie Cluster Plot with k=4 since there is too much overlap to analyze. Hollywood Movie Cluster Plot with k=3 would be the most optimal since it has more than 2 variables and has less overlap than Hollywood Movie Cluster Plot with k=4.

## Profitability Column

### Decision Tree Profitability

#### With Categorical Variables

This section and the next section are meant to be used as a comparison between the decision trees of Profitability that have categorical variables or not.

movie<- movie %>%  
 na.omit()  
movie

## # A tibble: 591 × 16  
## Movie LeadStudio RottenTomatoes AudienceScore Story Genre TheatersOpenWeek  
## <chr> <chr> <dbl> <dbl> <chr> <chr> <dbl>  
## 1 Spider-… Sony 61 54 Meta… Acti… 4252  
## 2 Shrek t… Paramount 42 57 Quest Anim… 4122  
## 3 Transfo… Paramount 57 89 Mons… Acti… 4011  
## 4 Pirates… Disney 45 74 Resc… Acti… 4362  
## 5 Harry P… Warner Br… 78 82 Quest Adve… 4285  
## 6 I Am Le… Warner Br… 69 69 Quest Thri… 3606  
## 7 The Bou… Universal 93 91 Purs… Thri… 3660  
## 8 Nationa… Disney 31 72 The … Thri… 3832  
## 9 Alvin a… Fox 26 73 Come… Anim… 3475  
## 10 300 Warner Br… 60 90 Sacr… Acti… 3103  
## # ℹ 581 more rows  
## # ℹ 9 more variables: OpeningWeekend <dbl>, BOAvgOpenWeekend <dbl>,  
## # DomesticGross <dbl>, ForeignGross <dbl>, WorldGross <dbl>, Budget <dbl>,  
## # Profitability <dbl>, OpenProfit <dbl>, Year <dbl>

The movie(HollywoodMovies dataset) dataset originally had many NA values in all columns, which needed to be removed. The code above is used to remove NA values with na.omit() since it needs to be removed to train the code and create decision and random trees.

HighProfit <- ifelse(movie$Profitability >= 400, "Yes", "No")  
  
movie<-data.frame(movie, HighProfit)  
  
movie$Profitability<-as.factor(movie$Profitability)

The code above creates a new variable called HighProfit that will be used to determine if a movie has a profit of more than 400 in the Profitability column. The HighProfit column will be determining this by labeling movies with string values Yes or No depending on the movie’s profitability.

#remove.packages("Rcpp")  
#install.packages('Rcpp')  
library(Rcpp)

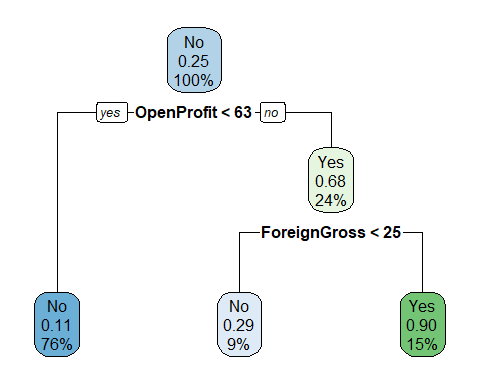
## Warning: package 'Rcpp' was built under R version 4.4.3

tree.movieProf<-train(HighProfit ~ . -Profitability, data=movie, method="rpart")  
tree.movieProf

## CART   
##   
## 591 samples  
## 16 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 591, 591, 591, 591, 591, 591, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.08163265 0.8425314 0.5445023  
## 0.14285714 0.8273639 0.4935204  
## 0.35374150 0.8092230 0.4007677  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.08163265.

The variable, HighProfit was trained with the dataset movie (the storedHollywood dataset) and without the variable, Profitability. As you can see, there are relatively low Kappa scores that suggest that the two raters do have a weak agreement with the two raters when using nominal scores. The most optimal model has an Accuracy of 0.8425314, cp of 0.08163265, and a Kappa of 0.5445023, which is not great because the Accuracy was average and the Kappa was low. This training method used 25 Bootstrapped reps for resampling.

rpart.plot(tree.movieProf$finalModel)



For this section, the categorical variable was not removed so that the decision trees with categorical variables and without categorical variables can be compared.

If OpenProfit < 63: the model predicts low profitability (< 400) with 11% confidence in “No”. 76% of the movies fall in this group.

If OpenProfit ≥ 63 and ForeignGross < 25: the model still predicts low profitability with 29% confidence in “No”. 9% of the movies fall in this group.

If OpenProfit ≥ 63 and ForeignGross ≥ 25: the model predicts high profitability (≥ 400) with 90% confidence in “Yes”. 15% fall in this group.

This decision tree is somewhat poor since the depth of the decision tree is too small and too few variables to consider. That is why the next section is the same as this section but the decision tree does not include the categorical variables.

These rules can be shown at the bottom.

rpart.rules(tree.movieProf$finalModel)

## .outcome   
## 0.11 when OpenProfit < 63   
## 0.29 when OpenProfit >= 63 & ForeignGross < 25  
## 0.90 when OpenProfit >= 63 & ForeignGross >= 25

#### Removed Categorical Variables

This section and the previous section are meant to be used as a comparison between the decision trees of Profitability that have categorical variables or not. This section removes the categorical variables, Movie and LeadStudio.

movie <- read\_csv("https://raw.githubusercontent.com/reisanar/datasets/master/HollywoodMovies.csv")

## Rows: 970 Columns: 16  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (4): Movie, LeadStudio, Story, Genre  
## dbl (12): RottenTomatoes, AudienceScore, TheatersOpenWeek, OpeningWeekend, B...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

myvars<-c("RottenTomatoes", "AudienceScore", "TheatersOpenWeek", "DomesticGross", "ForeignGross","WorldGross","Budget","Profitability","OpeningWeekend","BOAvgOpenWeekend")

movie<-movie[myvars] %>%  
 na.omit()  
  
movie<-data.frame(movie)  
  
movie<-movie %>%  
 mutate(HighProfit=ifelse(movie$Profitability >= 400, "Yes", "No")) %>%  
 dplyr::select(-Profitability)  
  
movie<-movie %>%  
 mutate(HighProfit=as.factor(HighProfit))

The movie dataset originally had many NA values in all columns, which need to be removed. The code above is used to remove NA values with na.omit() since it needs to be removed to train the code and create decision and random trees. The code above also creates a new variable called HighProfit that will be used to determine if a movie has a profit of more than 400 in the Profitability column. The HighProfit column will be determining this by labeling movies with string values of Yes or No depending on the movie’s profitability.

inTrain<-createDataPartition(y=movie$HighProfit, p=0.75, list=FALSE)  
train <- movie[inTrain,]  
test <- movie[-inTrain,]

set.seed(217)  
cvcontrol <- trainControl(method = "repeatedcv", number = 10, allowParallel = TRUE)

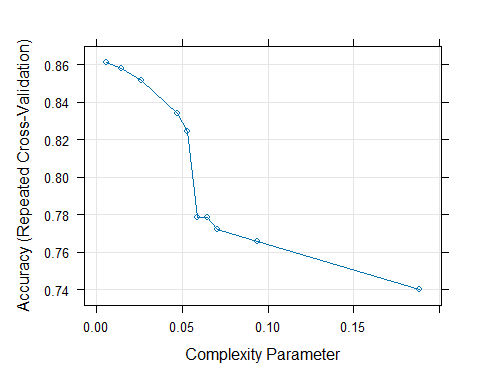
The two chunks above create training and testing sets for the HighProfit variable in the dataset, movie. The split is needed to train the data in order to better create decision trees. A trainControl variable was made to be used on the train variable shown at the bottom. The variable, cvcontril is made with the trainControl() method that uses the method repeatedcv, which is cv that repeats, 10-folds classification, and allows the code to run parallel.

dt\_movie<- train(HighProfit ~ . , data = train, method = "rpart",   
 trControl = cvcontrol, tuneLength = 10)  
dt\_movie

## CART   
##   
## 627 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 1 times)   
## Summary of sample sizes: 565, 564, 564, 564, 564, 565, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.005882353 0.8611623 0.6275904  
## 0.014705882 0.8579621 0.6152119  
## 0.026470588 0.8516385 0.6003964  
## 0.047058824 0.8340758 0.5565113  
## 0.052941176 0.8245520 0.5243331  
## 0.058823529 0.7782386 0.3981716  
## 0.064705882 0.7782386 0.3981716  
## 0.070588235 0.7718638 0.3816776  
## 0.094117647 0.7655146 0.3665162  
## 0.188235294 0.7400410 0.1125865  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.005882353.

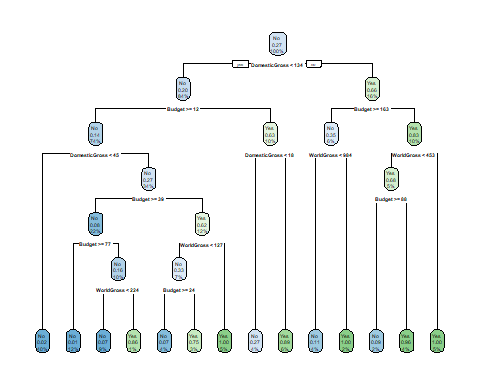
The train() method-used model uses a 10-fold classification done at 1 time and includes 2 classes that define if the movie is of HighProfit or not. The most optimal model, according to the CART table shown above, has a cp value of 0.005882353, accuracy of 0.8611623, and a Kappa of 0.6275904, which is average and not the absolute best in determining the best decision process to get the most profitable movie.

plot(dt\_movie)



The plot above shows the relationship between the complexity parameter and the Accuracy (Repeated Cross-Validation). The graph shows it has a negative relationship. This suggests that as the complexity parameter increases, the accuracy decreases.

rpart.plot(dt\_movie$finalModel)



The decision tree shown above does not include categorical variables that would hinder the creation of a precise decision tree. This is a much more complex decision tree compared to the previous section since it has more depth than the decision tree with categorical variables.

The best chance to have movies with Profitability of more than 400 million dollars with 100% confidence occurs in 3 scenarios:

If DomesticGross ≥ 45 & DomesticGross < 134, Budget ≥ 12 & Budget < 39, and WorldGross < 127: the model predicts high profitability (≥ 400) with 100% confidence in “Yes”. 5% of the movies fall in this group.

If DomesticGross ≥ 134, Budget ≥ 163, and WorldGross ≥ 984: the model predicts high profitability (≥ 400) with 100% confidence in “Yes”. 2% of the movies fall in this group.

If DomesticGross ≥ 134, Budget < 163, and WorldGross ≥ 453: the model predicts high profitability (≥ 400) with 100% confidence in “Yes”. 5% of the movies fall in this group.

However, the Yes leaf nodes have high error rates, so this decision tree is not an ideal model to predict a movie’s chance of being highly profitable.

tree\_classTrain<-predict(dt\_movie, type="raw")  
head(tree\_classTrain)

## [1] No Yes Yes No Yes Yes  
## Levels: No Yes

confusionMatrix(train$HighProfit, tree\_classTrain)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 447 10  
## Yes 24 146  
##   
## Accuracy : 0.9458   
## 95% CI : (0.925, 0.9622)  
## No Information Rate : 0.7512   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.8592   
##   
## Mcnemar's Test P-Value : 0.02578   
##   
## Sensitivity : 0.9490   
## Specificity : 0.9359   
## Pos Pred Value : 0.9781   
## Neg Pred Value : 0.8588   
## Prevalence : 0.7512   
## Detection Rate : 0.7129   
## Detection Prevalence : 0.7289   
## Balanced Accuracy : 0.9425   
##   
## 'Positive' Class : No   
##

The accuracy of the confusion matrix and Statistics is relatively great since it is 0.9458 correct. The 95% confidence interval falls in this range, (0.925, 0.9622). The kappa is relatively good and has a value of 0.8592, which means that there is a somewhat near-perfect agreement. This confusion matrix also claims that the p-value is less than 2e-16 which shows that the model does go against the null hypothesis.

tree\_classTest<-predict(dt\_movie, newdata=test, type="prob")  
tree\_classTest

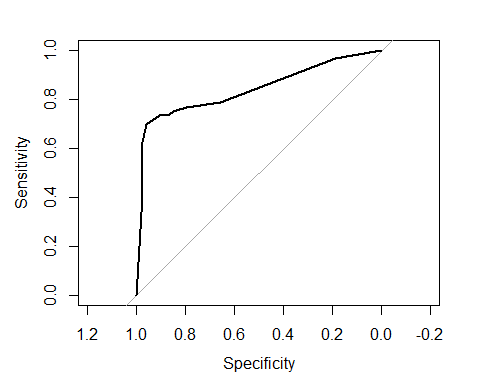
## No Yes  
## 6 0.00000000 1.00000000  
## 7 0.90909091 0.09090909  
## 8 0.00000000 1.00000000  
## 14 0.04347826 0.95652174  
## 16 0.90909091 0.09090909  
## 21 0.98648649 0.01351351  
## 22 0.00000000 1.00000000  
## 23 0.98648649 0.01351351  
## 35 0.00000000 1.00000000  
## 36 0.98648649 0.01351351  
## 40 0.98648649 0.01351351  
## 41 0.25000000 0.75000000  
## 43 0.92592593 0.07407407  
## 44 0.00000000 1.00000000  
## 47 0.92592593 0.07407407  
## 54 0.97609562 0.02390438  
## 55 0.97609562 0.02390438  
## 56 0.97609562 0.02390438  
## 58 0.97609562 0.02390438  
## 62 0.97609562 0.02390438  
## 65 0.97609562 0.02390438  
## 66 0.97609562 0.02390438  
## 77 0.97609562 0.02390438  
## 81 0.97609562 0.02390438  
## 83 0.97609562 0.02390438  
## 86 0.97609562 0.02390438  
## 87 0.97609562 0.02390438  
## 97 0.00000000 1.00000000  
## 100 0.04347826 0.95652174  
## 109 0.98648649 0.01351351  
## 112 0.14285714 0.85714286  
## 115 0.98648649 0.01351351  
## 121 0.92727273 0.07272727  
## 126 0.98648649 0.01351351  
## 128 0.98648649 0.01351351  
## 129 0.98648649 0.01351351  
## 141 0.11111111 0.88888889  
## 142 0.92727273 0.07272727  
## 145 0.92592593 0.07407407  
## 148 0.98648649 0.01351351  
## 151 0.97609562 0.02390438  
## 157 0.97609562 0.02390438  
## 161 0.97609562 0.02390438  
## 163 0.97609562 0.02390438  
## 169 0.97609562 0.02390438  
## 171 0.97609562 0.02390438  
## 175 0.97609562 0.02390438  
## 177 0.97609562 0.02390438  
## 180 0.97609562 0.02390438  
## 187 0.97609562 0.02390438  
## 201 0.97609562 0.02390438  
## 203 0.97609562 0.02390438  
## 209 0.73076923 0.26923077  
## 215 0.97609562 0.02390438  
## 222 0.97609562 0.02390438  
## 224 0.88888889 0.11111111  
## 229 0.90909091 0.09090909  
## 232 0.00000000 1.00000000  
## 243 0.88888889 0.11111111  
## 248 0.98648649 0.01351351  
## 253 0.98648649 0.01351351  
## 258 0.00000000 1.00000000  
## 261 0.98648649 0.01351351  
## 263 0.92727273 0.07272727  
## 264 0.92727273 0.07272727  
## 269 0.92727273 0.07272727  
## 271 0.98648649 0.01351351  
## 272 0.25000000 0.75000000  
## 279 0.92727273 0.07272727  
## 284 0.92592593 0.07407407  
## 285 0.97609562 0.02390438  
## 287 0.97609562 0.02390438  
## 288 0.97609562 0.02390438  
## 296 0.97609562 0.02390438  
## 300 0.97609562 0.02390438  
## 301 0.97609562 0.02390438  
## 303 0.11111111 0.88888889  
## 305 0.97609562 0.02390438  
## 309 0.97609562 0.02390438  
## 317 0.97609562 0.02390438  
## 329 0.97609562 0.02390438  
## 333 0.97609562 0.02390438  
## 339 0.73076923 0.26923077  
## 346 0.97609562 0.02390438  
## 350 0.97609562 0.02390438  
## 351 0.97609562 0.02390438  
## 356 0.97609562 0.02390438  
## 358 0.97609562 0.02390438  
## 366 0.97609562 0.02390438  
## 367 0.97609562 0.02390438  
## 371 0.00000000 1.00000000  
## 378 0.92592593 0.07407407  
## 379 0.97609562 0.02390438  
## 380 0.00000000 1.00000000  
## 386 0.92727273 0.07272727  
## 392 0.97609562 0.02390438  
## 398 0.00000000 1.00000000  
## 412 0.92592593 0.07407407  
## 416 0.73076923 0.26923077  
## 421 0.97609562 0.02390438  
## 424 0.97609562 0.02390438  
## 426 0.98648649 0.01351351  
## 446 0.97609562 0.02390438  
## 448 0.98648649 0.01351351  
## 449 0.92727273 0.07272727  
## 465 0.14285714 0.85714286  
## 467 0.97609562 0.02390438  
## 478 0.97609562 0.02390438  
## 486 0.97609562 0.02390438  
## 492 0.92727273 0.07272727  
## 498 0.90909091 0.09090909  
## 505 0.25000000 0.75000000  
## 510 0.97609562 0.02390438  
## 513 0.97609562 0.02390438  
## 515 0.00000000 1.00000000  
## 526 0.92727273 0.07272727  
## 528 0.97609562 0.02390438  
## 531 0.98648649 0.01351351  
## 535 0.00000000 1.00000000  
## 536 0.97609562 0.02390438  
## 545 0.92727273 0.07272727  
## 548 0.11111111 0.88888889  
## 554 0.98648649 0.01351351  
## 559 0.97609562 0.02390438  
## 560 0.97609562 0.02390438  
## 561 0.73076923 0.26923077  
## 566 0.97609562 0.02390438  
## 572 0.97609562 0.02390438  
## 575 0.97609562 0.02390438  
## 578 0.00000000 1.00000000  
## 579 0.92727273 0.07272727  
## 584 0.04347826 0.95652174  
## 585 0.97609562 0.02390438  
## 587 0.92727273 0.07272727  
## 588 0.97609562 0.02390438  
## 600 0.92727273 0.07272727  
## 604 0.92727273 0.07272727  
## 608 0.97609562 0.02390438  
## 611 0.97609562 0.02390438  
## 615 0.14285714 0.85714286  
## 621 0.88888889 0.11111111  
## 623 0.25000000 0.75000000  
## 625 0.92592593 0.07407407  
## 627 0.25000000 0.75000000  
## 632 0.00000000 1.00000000  
## 635 0.97609562 0.02390438  
## 642 0.98648649 0.01351351  
## 644 0.97609562 0.02390438  
## 646 0.04347826 0.95652174  
## 647 0.98648649 0.01351351  
## 648 0.04347826 0.95652174  
## 652 0.97609562 0.02390438  
## 654 0.88888889 0.11111111  
## 657 0.11111111 0.88888889  
## 664 0.97609562 0.02390438  
## 668 0.98648649 0.01351351  
## 673 0.11111111 0.88888889  
## 674 0.00000000 1.00000000  
## 676 0.97609562 0.02390438  
## 678 0.04347826 0.95652174  
## 679 0.00000000 1.00000000  
## 683 0.98648649 0.01351351  
## 694 0.97609562 0.02390438  
## 701 0.00000000 1.00000000  
## 703 0.97609562 0.02390438  
## 705 0.92592593 0.07407407  
## 708 0.97609562 0.02390438  
## 709 0.98648649 0.01351351  
## 710 0.88888889 0.11111111  
## 711 0.92727273 0.07272727  
## 712 0.97609562 0.02390438  
## 713 0.97609562 0.02390438  
## 716 0.98648649 0.01351351  
## 717 0.92727273 0.07272727  
## 727 0.00000000 1.00000000  
## 731 0.88888889 0.11111111  
## 743 0.98648649 0.01351351  
## 748 0.98648649 0.01351351  
## 749 0.04347826 0.95652174  
## 750 0.98648649 0.01351351  
## 751 0.98648649 0.01351351  
## 753 0.04347826 0.95652174  
## 755 0.92727273 0.07272727  
## 756 0.98648649 0.01351351  
## 761 0.98648649 0.01351351  
## 763 0.92727273 0.07272727  
## 767 0.00000000 1.00000000  
## 768 0.00000000 1.00000000  
## 769 0.98648649 0.01351351  
## 770 0.92727273 0.07272727  
## 771 0.00000000 1.00000000  
## 777 0.92727273 0.07272727  
## 780 0.97609562 0.02390438  
## 783 0.92727273 0.07272727  
## 787 0.92592593 0.07407407  
## 794 0.97609562 0.02390438  
## 795 0.25000000 0.75000000  
## 797 0.97609562 0.02390438  
## 801 0.11111111 0.88888889  
## 808 0.97609562 0.02390438  
## 811 0.97609562 0.02390438  
## 814 0.97609562 0.02390438  
## 818 0.73076923 0.26923077  
## 823 0.73076923 0.26923077  
## 826 0.73076923 0.26923077  
## 828 0.73076923 0.26923077  
## 830 0.73076923 0.26923077  
## 832 0.73076923 0.26923077

ROC\_curve\_dt<-roc(test$HighProfit, tree\_classTest[,"Yes"])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(ROC\_curve\_dt)



auc(ROC\_curve\_dt)

## Area under the curve: 0.8392

The graph shown above shows that the Area under the curve takes up 0.8392 which means that the performance of the model poorly distinguishes between the positive and negative classes. This graph will be saved in order to compare how well all the decision tree techniques interpreted the test data.

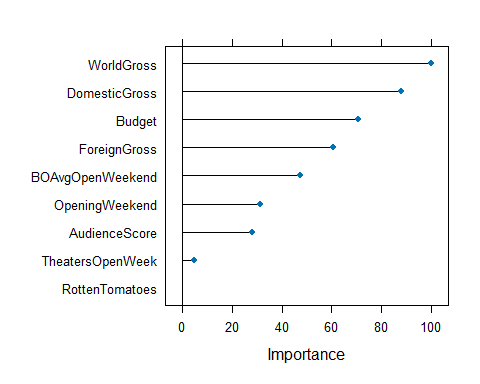
### Bagging

bag\_movie<-train(HighProfit~.,data=train, method="treebag", trControl=cvcontrol, importance=TRUE)  
  
bag\_movie

## Bagged CART   
##   
## 627 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 1 times)   
## Summary of sample sizes: 564, 564, 565, 564, 565, 564, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.9232719 0.7976616

The bagged CART model has two predictors that determine if the movie has a high profit of more than 400 million dollars. The bagging will be used to improve the accuracy of the algorithms used in the machine learning algorithm. The bag\_movie contains 2 classes called Yes and No that indicate if the movie agrees with HighProfit. It also has only one optimal model with an accuracy of 0.9232719 and a kappa value of 0.7976616.

plot(varImp(bag\_movie))



The varImpplot shown above shows that WorldGross, DomesticGross, Budget, and ForeignGross are the most important to the bagged HighProfit variable with importance values of 100, 90, 70, and 60. This means that the project should consider these variables as significant to the training and testing of the variable HighProfit.

bagg\_pred<-predict(bag\_movie, newdata = test, type="raw")  
head(bagg\_pred)

## [1] Yes Yes No Yes No No   
## Levels: No Yes

confusionMatrix(test$HighProfit, bagg\_pred)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 146 6  
## Yes 15 41  
##   
## Accuracy : 0.899   
## 95% CI : (0.8498, 0.9364)  
## No Information Rate : 0.774   
## P-Value [Acc > NIR] : 2.25e-06   
##   
## Kappa : 0.7297   
##   
## Mcnemar's Test P-Value : 0.08086   
##   
## Sensitivity : 0.9068   
## Specificity : 0.8723   
## Pos Pred Value : 0.9605   
## Neg Pred Value : 0.7321   
## Prevalence : 0.7740   
## Detection Rate : 0.7019   
## Detection Prevalence : 0.7308   
## Balanced Accuracy : 0.8896   
##   
## 'Positive' Class : No   
##

The accuracy of the confusion matrix and Statistics is relatively good since it is 0.899. The 95% confidence interval has a range of (0.8498, 0.9364). The kappa is relatively average with a value of 0.7297, which is not very impressive. This confusion matrix also claims that the p-value is 2.25e-06 which is very low and that this model does go against the null hypothesis. The confusion matrix and Statistics does well because of the decent Accuracy and moderate Kappa.

bagg\_probs <- predict(bag\_movie, newdata = test, type = "prob")  
head(bagg\_probs)

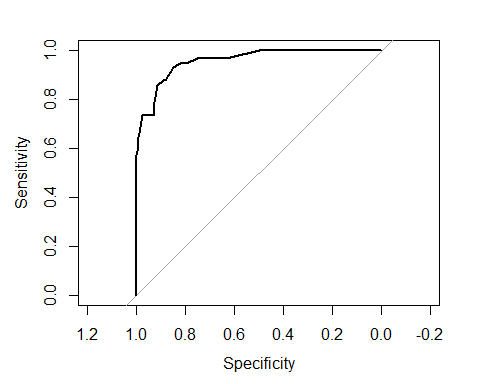
## No Yes  
## 1 0.48 0.52  
## 2 0.44 0.56  
## 3 0.52 0.48  
## 4 0.04 0.96  
## 5 0.96 0.04  
## 6 1.00 0.00

ROC\_curve\_bagg <- roc(test$HighProfit, bagg\_probs[ , "Yes"])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(ROC\_curve\_bagg)



auc(ROC\_curve\_bagg)

## Area under the curve: 0.9571

The graph shown above shows that the Area under the curve takes up 95.71% which means that the performance of the model can easily distinguish between the positive and negative classes. This graph will be saved in order to compare how well all the decision tree techniques interpreted the test data.

### Random Forest

rf\_HighProfit<-train(HighProfit~., data=train, method="rf", trControl=cvcontrol, importance =TRUE)  
rf\_HighProfit

## Random Forest   
##   
## 627 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 1 times)   
## Summary of sample sizes: 564, 565, 564, 564, 564, 565, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.9074245 0.7547333  
## 5 0.9266769 0.8095847  
## 9 0.9346646 0.8298408  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 9.

The training method for the random forest has 2 classes that determine if HighProfit is high by 400 million dollars or more. This training method was also done in a 10-fold classification repeated 1 time. The random forest method will be used to improve the accuracy of the algorithms used in the machine learning algorithm. It contains 2 classes called Yes and No that indicate if the movie agrees with HighProfit. It also has an optimal model with an accuracy of 0.9346646, a kappa value of 0.8298408, and a mtry of 9. This random forest is relatively above average since the Accuracy is high and the Kappa is average.

rf\_pred<-predict(rf\_HighProfit, newdata=test, type="raw")  
confusionMatrix(test$HighProfit, rf\_pred)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 149 3  
## Yes 14 42  
##   
## Accuracy : 0.9183   
## 95% CI : (0.8724, 0.9517)  
## No Information Rate : 0.7837   
## P-Value [Acc > NIR] : 1.592e-07   
##   
## Kappa : 0.7786   
##   
## Mcnemar's Test P-Value : 0.01529   
##   
## Sensitivity : 0.9141   
## Specificity : 0.9333   
## Pos Pred Value : 0.9803   
## Neg Pred Value : 0.7500   
## Prevalence : 0.7837   
## Detection Rate : 0.7163   
## Detection Prevalence : 0.7308   
## Balanced Accuracy : 0.9237   
##   
## 'Positive' Class : No   
##

The accuracy of the confusion matrix and Statistics is relatively great since it is 91.83%. The 95% confidence interval has a range of (0.8724, 0.9517). The kappa is very mediocre with a value of 0.7786, which means that there is some agreement. This confusion matrix also claims that the p-value is 1.592e-07 which is very low and that this model does greatly go against the null hypothesis. The confusion matrix and Statistics is relatively great due to the high Accuracy and the low P-value.

rf\_probs<-predict(rf\_HighProfit, newdata = test, type = "prob")  
head(rf\_probs)

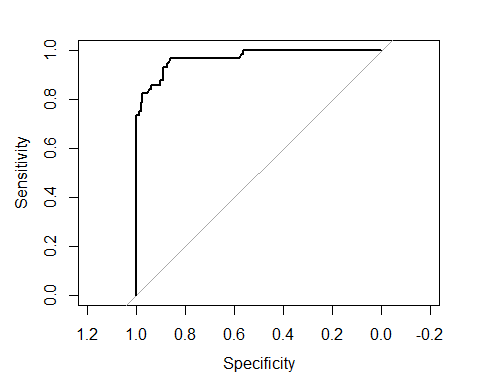
## No Yes  
## 6 0.464 0.536  
## 7 0.536 0.464  
## 8 0.634 0.366  
## 14 0.002 0.998  
## 16 0.942 0.058  
## 21 0.986 0.014

ROC\_curve\_rf<-roc(test$HighProfit, rf\_probs[,"Yes"])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(ROC\_curve\_rf)



auc(ROC\_curve\_rf)

## Area under the curve: 0.9685

The graph shown above shows that the Area under the curve takes up 96.85% which means that the performance of the model can easily distinguish between the positive and negative classes which are almost perfect. This graph will be saved in order to compare how well all the decision tree techniques interpreted the test data. The AUC of ROC\_curve\_rf is relatively higher compared to the past 2 methods which means that this is more useful than the other two methods since its overall performance would be better.

### Boosting

gbm\_movie<-train(HighProfit~., data=train,  
 method="gbm", verbose=F, trControl=cvcontrol)  
gbm\_movie

## Stochastic Gradient Boosting   
##   
## 627 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 1 times)   
## Summary of sample sizes: 564, 565, 565, 565, 564, 564, ...   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.8166923 0.4630568  
## 1 100 0.8581669 0.6083219  
## 1 150 0.8900922 0.7075952  
## 2 50 0.8836150 0.6808394  
## 2 100 0.9155146 0.7735152  
## 2 150 0.9315412 0.8211607  
## 3 50 0.9107527 0.7591800  
## 3 100 0.9346902 0.8287538  
## 3 150 0.9410394 0.8487608  
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
##   
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 150, interaction.depth =  
## 3, shrinkage = 0.1 and n.minobsinnode = 10.

The Stochastic Gradient Boosting with 10-fold cross-validation shows the most optimal model has 150 trees, an interactive depth of 3, shrinkage of 0.1, Accuracy of 0.9410394, and Kappa of 0.8487608, which can be said as a relatively great model because of its Kappa and Accuracy. This is done to the variable, HighProfit, since the project needs to determine what makes a movie highly profitable.

gbm\_preds <- predict(gbm\_movie, newdata = test, type = "raw")  
# get confusion matrix  
confusionMatrix(test$HighProfit, gbm\_preds)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 148 4  
## Yes 12 44  
##   
## Accuracy : 0.9231   
## 95% CI : (0.8781, 0.9554)  
## No Information Rate : 0.7692   
## P-Value [Acc > NIR] : 3.816e-09   
##   
## Kappa : 0.7953   
##   
## Mcnemar's Test P-Value : 0.08012   
##   
## Sensitivity : 0.9250   
## Specificity : 0.9167   
## Pos Pred Value : 0.9737   
## Neg Pred Value : 0.7857   
## Prevalence : 0.7692   
## Detection Rate : 0.7115   
## Detection Prevalence : 0.7308   
## Balanced Accuracy : 0.9208   
##   
## 'Positive' Class : No   
##

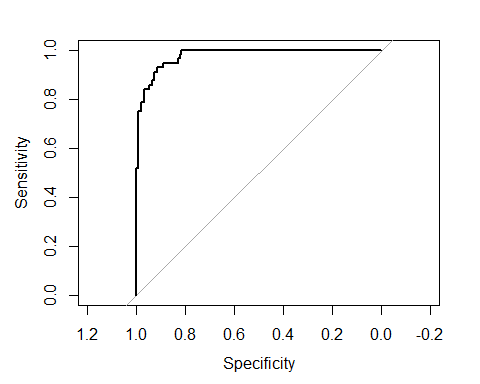
The confusionMatrix and Statistics for this section shows that the accuracy of 0.9231, the 95% confidence interval of (0.8781, 0.9554), Kappa value of 0.7953, and the p-value of 3.816e-09, which means that the model does well and it goes against the null hypothesis.

gbm\_probs<-predict(gbm\_movie, newdata = test, type = "prob")  
ROC\_curve\_gbm<-roc(test$HighProfit, gbm\_probs[,"Yes"])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(ROC\_curve\_gbm)



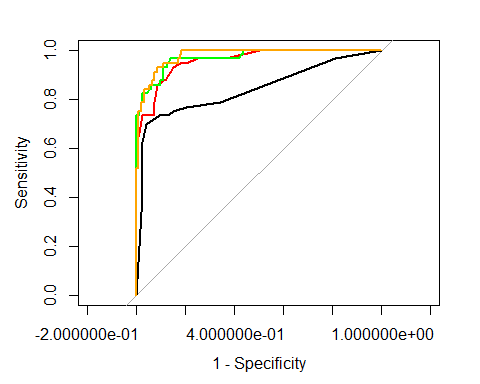
auc(ROC\_curve\_gbm)

## Area under the curve: 0.9783

The graph shown above shows that the Area under the curve takes up 97.83%, which means that the performance of the model can easily distinguish between the positive and negative classes and has a high performance. This graph will be saved in order to compare how well all the decision tree techniques interpreted the test data. The AUC-ROC curve is relatively higher compared to the past 3 methods which means that this is more useful than the other two methods since its overall performance is better.

### Model Comparison High-Profitability

# using the `add = TRUE` option for each plot after the first one:  
plot(ROC\_curve\_dt, col = "black", legacy.axes = T, ) # black  
plot(ROC\_curve\_bagg, add = TRUE, col = "red") # color red for bagging  
plot(ROC\_curve\_rf, add = TRUE, col = "green") # color green is for random forest  
plot(ROC\_curve\_gbm,add = TRUE, col = "orange") # color orange is for boosting



auc(ROC\_curve\_dt)

## Area under the curve: 0.8392

auc(ROC\_curve\_bagg)

## Area under the curve: 0.9571

auc(ROC\_curve\_rf)

## Area under the curve: 0.9685

auc(ROC\_curve\_gbm)

## Area under the curve: 0.9783

The graphs shown above show the AUC-ROC scores of the decision tree, bagging, boosting, and rain forest models. The best method for the variable, HighProfit, is boosting, which is shown with the orange curve, since it has the highest AUC-ROC by a small margin, which we will consider highly. The worst method for the variable, HighProfit is decision tree since it has the lowest AUC-ROC by a large margin, which we will not consider.

## AudienceScore Column

### Decision Tree AudienceScore

#### With Categorical Variables

This section and the next section are meant to be used as a comparison between the decision trees of AudienceScore that have categorical variables or not.

movie <- read\_csv("https://raw.githubusercontent.com/reisanar/datasets/master/HollywoodMovies.csv")

## Rows: 970 Columns: 16  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (4): Movie, LeadStudio, Story, Genre  
## dbl (12): RottenTomatoes, AudienceScore, TheatersOpenWeek, OpeningWeekend, B...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

movie<- movie %>%  
 na.omit()  
movie

## # A tibble: 591 × 16  
## Movie LeadStudio RottenTomatoes AudienceScore Story Genre TheatersOpenWeek  
## <chr> <chr> <dbl> <dbl> <chr> <chr> <dbl>  
## 1 Spider-… Sony 61 54 Meta… Acti… 4252  
## 2 Shrek t… Paramount 42 57 Quest Anim… 4122  
## 3 Transfo… Paramount 57 89 Mons… Acti… 4011  
## 4 Pirates… Disney 45 74 Resc… Acti… 4362  
## 5 Harry P… Warner Br… 78 82 Quest Adve… 4285  
## 6 I Am Le… Warner Br… 69 69 Quest Thri… 3606  
## 7 The Bou… Universal 93 91 Purs… Thri… 3660  
## 8 Nationa… Disney 31 72 The … Thri… 3832  
## 9 Alvin a… Fox 26 73 Come… Anim… 3475  
## 10 300 Warner Br… 60 90 Sacr… Acti… 3103  
## # ℹ 581 more rows  
## # ℹ 9 more variables: OpeningWeekend <dbl>, BOAvgOpenWeekend <dbl>,  
## # DomesticGross <dbl>, ForeignGross <dbl>, WorldGross <dbl>, Budget <dbl>,  
## # Profitability <dbl>, OpenProfit <dbl>, Year <dbl>

The movie (HollywoodMovies dataset) dataset originally had many NA values in all columns, which needed to be removed. The code above is used to remove NA values with na.omit() since it needs to be removed in order to train the code and create decision and random trees.

HighAud <- ifelse(movie$AudienceScore >= 73, "Yes", "No")  
  
movie<-data.frame(movie, HighAud)  
  
movie$AudienceScore<-as.factor(movie$AudienceScore)

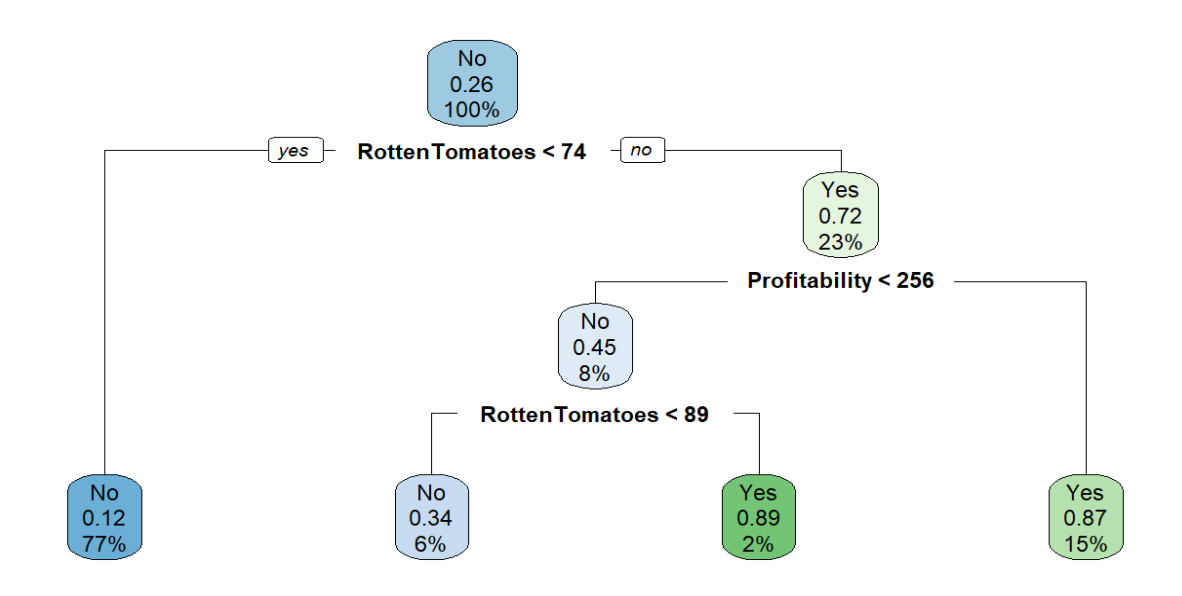
The code above creates a new variable called HighAud that will be used to determine if a movie has an audience rating of at least 73 in the AudienceScore column. The HighAud column will be determining this by labeling movies with string values Yes or No depending on the movie’s audience score.

tree.movieAud<-train(HighAud ~ . -AudienceScore, data=movie, method="rpart")  
tree.movieAud

## CART   
##   
## 591 samples  
## 16 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 591, 591, 591, 591, 591, 591, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.03571429 0.8360114 0.5552819  
## 0.03896104 0.8332730 0.5458896  
## 0.38961039 0.7918665 0.3172508  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.03571429.

The CART training method model uses 25 bootstraped reps for resampling. The variable, HighAud was trained with the dataset movie (the storedHollywood dataset) and without the variable, AudienceScore. As you can see that there are relatively low Kappa scores that suggest that the two raters do have a weak agreement with the two raters when using nominal scores. The most optimal model has an Accuracy of 0.8360114, cp of 0.03571429, and a Kappa of 0.5552819, which is not a very good CART model to use.

rpart.plot(tree.movieAud$finalModel)



For this section, the categorical variable was not removed so the decision trees with categorical variables and without categorical variables can be compared with this section and the next section.

If RottenTomatoes < 74: the model predicts low AudienceScore (< 73) with 12% confidence in “No”. 77% of the movies fall in this group.

If RottenTomatoes ≥ 74 and RottenTomatoes < 89 and Profitability < 256: the model predicts low AudienceScore (< 73) with 34% confidence in “No”. 6% of the movies fall in this group.

If RottenTomatoes ≥ 89 and Profitability < 256: the model predicts high AudienceScore (≥ 73) with 89% confidence in “Yes”. 2% of the movies fall in this group.

If RottenTomatoes ≥ 74 and Profitability ≥ 256: the model predicts high AudienceScore (≥ 73) with 87% confidence in “Yes”. 15% of the movies fall in this group.

These rules are shown at the bottom.

rpart.rules(tree.movieAud$finalModel)

## .outcome   
## 0.12 when RottenTomatoes < 74  
## 0.34 when RottenTomatoes >= 74 to 89 & Profitability < 256 ## 0.87 when RottenTomatoes >= 74 & Profitability >= 256 ## 0.89 when RottenTomatoes >= 89 & Profitability < 256

#### Removed Categorical Variables

This section and the previous section are meant to be used as a comparison between the decision trees of AudienceScore that have categorical variables or not. This section removes the categorical variables, Movie and LeadStudio.

movie <- read\_csv("https://raw.githubusercontent.com/reisanar/datasets/master/HollywoodMovies.csv")

## Rows: 970 Columns: 16  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (4): Movie, LeadStudio, Story, Genre  
## dbl (12): RottenTomatoes, AudienceScore, TheatersOpenWeek, OpeningWeekend, B...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

myvars<-c("RottenTomatoes", "AudienceScore", "TheatersOpenWeek", "DomesticGross", "ForeignGross","WorldGross","Budget","Profitability","OpeningWeekend","BOAvgOpenWeekend")

movie<-movie[myvars] %>%  
 na.omit()  
  
movie<-data.frame(movie)  
  
movie<-movie %>%  
 mutate(HighAud=ifelse(movie$AudienceScore >= 73, "Yes", "No")) %>%  
 dplyr::select(-AudienceScore)  
  
movie<-movie %>%  
 mutate(HighAud=as.factor(HighAud))

The movie dataset originally had many NA values in all columns, which need to be removed. The code above is used to remove NA values with na.omit() since it needs to be removed to train the code and create a decision and random trees. The code above also creates a new variable called HighAud that will be used to determine if a movie has an audience score of more than 73 in the AudienceScore column. The HighAud column will be determining this by labeling movies with string values of Yes or No depending on the movie’s audience reviews.

inTrain<-createDataPartition(y=movie$HighAud, p=0.75, list=FALSE)  
train <- movie[inTrain,]  
test <- movie[-inTrain,]

set.seed(217)  
cvcontrol <- trainControl(method = "repeatedcv", number = 10, allowParallel = TRUE)

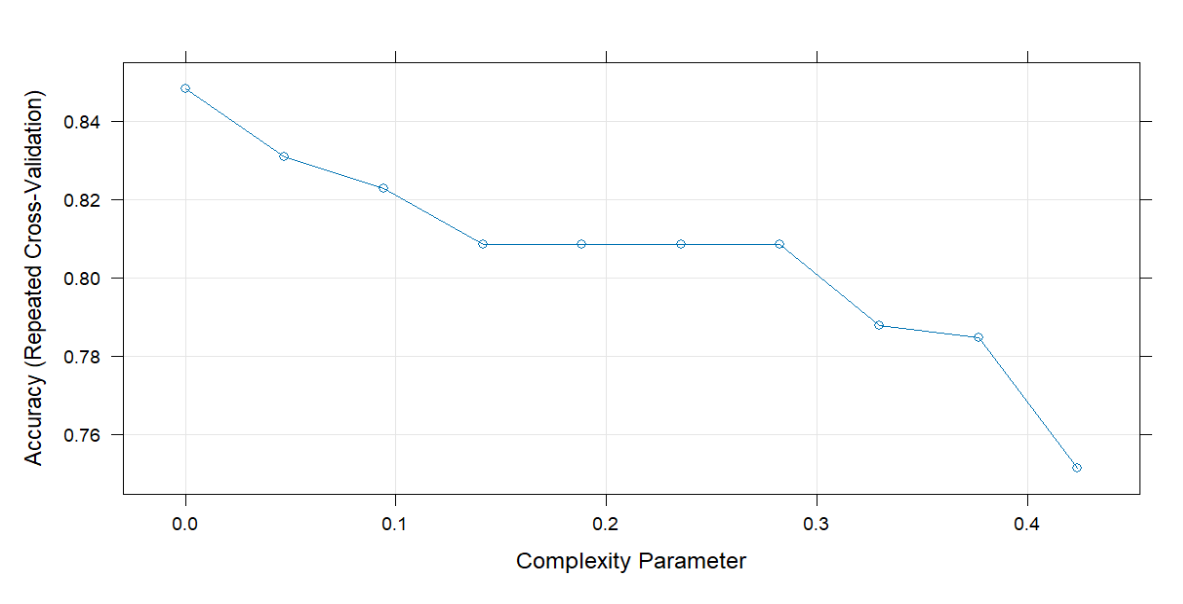
The two chunks above create the training and testing sets for the HighAud variable in the dateset, movie. The split is needed to train the data to better create decision trees. A trainControl variable was made to be used on the train variable shown at the bottom.

dt\_movie<- train(HighAud ~ . , data = train, method = "rpart",   
 trControl = cvcontrol, tuneLength = 10)  
dt\_movie

## CART   
##   
## 627 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 1 times)   
## Summary of sample sizes: 565, 564, 565, 565, 564, 564, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.00000000 0.8482847 0.6227430  
## 0.05335844 0.8500512 0.6134471  
## 0.10671689 0.8500512 0.6134471  
## 0.16007533 0.8500512 0.6134471  
## 0.21343377 0.8500512 0.6134471  
## 0.26679222 0.8500512 0.6134471  
## 0.32015066 0.8500512 0.6134471  
## 0.37350910 0.8500512 0.6134471  
## 0.42686755 0.8500512 0.6134471  
## 0.48022599 0.7878904 0.3524136  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.

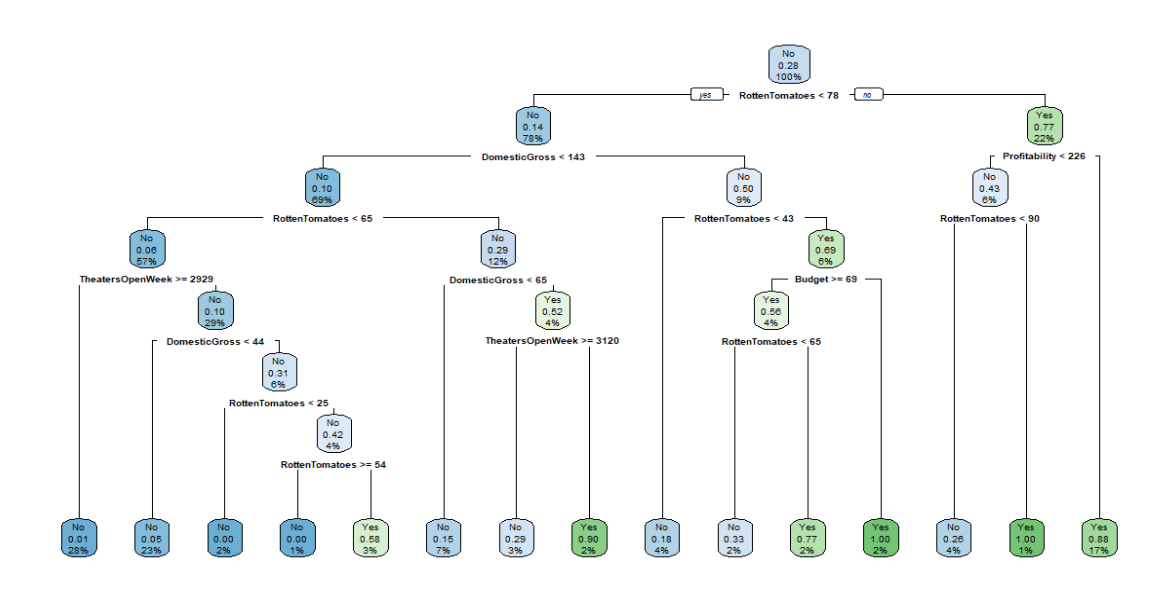
The CART model uses the HighAud variable and is resampled with 10-fold classification and cross-validation at one time. The most optimal model, according to the CART table shown above, has a cp value of 0, accuracy of 0.8482847, and a Kappa of 0.6227430, which is not great since the Accuracy is mediocre, and the Kappa is low.

plot(dt\_movie)



The plot above shows the relationship between the complexity parameter and the Accuracy (Repeated Cross-Validation).The graph has a consistent decline in accuracy as the complexity parameter increases.

rpart.plot(dt\_movie$finalModel)



The decision tree shown above does not include categorical variables that would hinder the creation of a precise decision tree. This is a much superior decision tree compared to the previous section since it has more depth than the decision tree with categorical variables.

The best chance to find movies with a high AudienceScore of more than 73 with 100% confidence occur in 2 scenarios:

If RottenTomatoes ≥ 65 and RottenTomatoes < 78, DomesticGross ≥ 143, and Budget < 69: the model predicts high AudienceScore (≥ 73) with 100% confidence in “Yes”. 2% of the movies fall in this group.

If RottenTomatoes ≥ 90 and Profitability < 226: the model predicts high AudienceScore (≥ 73) with 100% confidence in “Yes”. 1% of the movies fall in this group.

tree\_classTrain<-predict(dt\_movie, type="raw")  
head(tree\_classTrain)

## [1] No No No Yes Yes Yes   
## Levels: No Yes

confusionMatrix(train$HighAud, tree\_classTrain)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 426 24  
## Yes 36 141  
##   
## Accuracy : 09043  
## 95% CI : (0.8785, 0.9262)  
## No Information Rate : 0.7368   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.7589   
##   
## Mcnemar's Test P-Value : 0.1156   
##   
## Sensitivity : 0.9221   
## Specificity : 0.8545   
## Pos Pred Value : 0.9467   
## Neg Pred Value : 0.7966   
## Prevalence : 0.7368   
## Detection Rate : 0.6794   
## Detection Prevalence : 0.7177   
## Balanced Accuracy : 0.8883   
##   
## 'Positive' Class : No   
##

The confusionMatrix and Statistics for this section shows that the accuracy of 0.9043, the 95% confidence interval of (0.8785, 0.9262), Kappa value of 0.7589, and the p-value of less than 2e-16, which means that it goes against the null hypothesis. This model is average since its Kappa value indicates the overall agreement level is mediocre.

tree\_classTest<-predict(dt\_movie, newdata=test, type="prob")  
tree\_classTest

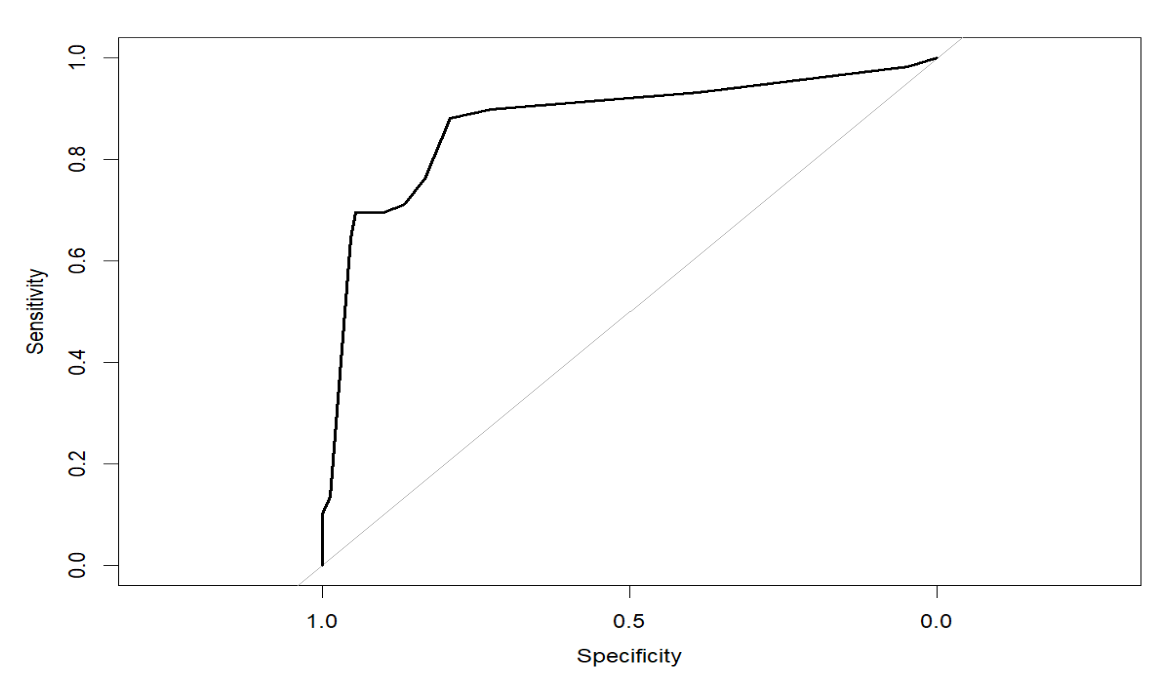
## No Yes  
## 1 0.8744770 0.1255230  
## 2 0.8744770 0.1255230  
## 4 0.8744770 0.1255230  
## 6 0.8744770 0.1255230  
## 14 0.2147651 0.7852349  
## 18 0.8744770 0.1255230  
## 23 0.8744770 0.1255230  
## 27 0.8744770 0.1255230  
## 30 0.8744770 0.1255230  
## 31 0.8744770 0.1255230  
## 36 0.8744770 0.1255230  
## 42 0.8744770 0.1255230  
## 45 0.2147651 0.7852349  
## 52 0.8744770 0.1255230  
## 55 0.8744770 0.1255230  
## 61 0.8744770 0.1255230  
## 62 0.8744770 0.1255230  
## 65 0.8744770 0.1255230  
## 67 0.8744770 0.1255230  
## 79 0.8744770 0.1255230  
## 81 0.8744770 0.1255230  
## 84 0.8744770 0.1255230  
## 85 0.8744770 0.1255230  
## 86 0.8744770 0.1255230  
## 90 0.8744770 0.1255230  
## 95 0.2147651 0.7852349  
## 98 0.8744770 0.1255230  
## 103 0.8744770 0.1255230  
## 111 0.8744770 0.1255230  
## 118 0.8744770 0.1255230  
## 122 0.8744770 0.1255230  
## 127 0.8744770 0.1255230  
## 139 0.8744770 0.1255230  
## 142 0.8744770 0.1255230  
## 143 0.8744770 0.1255230  
## 144 0.8744770 0.1255230  
## 146 0.8744770 0.1255230  
## 147 0.8744770 0.1255230  
## 149 0.8744770 0.1255230  
## 151 0.8744770 0.1255230  
## 154 0.8744770 0.1255230  
## 155 0.8744770 0.1255230  
## 157 0.8744770 0.1255230  
## 160 0.8744770 0.1255230  
## 162 0.8744770 0.1255230  
## 167 0.8744770 0.1255230  
## 169 0.8744770 0.1255230  
## 170 0.8744770 0.1255230  
## 171 0.2147651 0.7852349  
## 173 0.8744770 0.1255230  
## 174 0.8744770 0.1255230  
## 177 0.8744770 0.1255230  
## 183 0.8744770 0.1255230  
## 188 0.8744770 0.1255230  
## 194 0.8744770 0.1255230  
## 195 0.2147651 0.7852349  
## 196 0.8744770 0.1255230  
## 197 0.2147651 0.7852349  
## 208 0.2147651 0.7852349  
## 227 0.2147651 0.7852349  
## 228 0.2147651 0.7852349  
## 230 0.8744770 0.1255230  
## 231 0.8744770 0.1255230  
## 232 0.8744770 0.1255230  
## 233 0.8744770 0.1255230  
## 234 0.8744770 0.1255230  
## 236 0.8744770 0.1255230  
## 238 0.8744770 0.1255230  
## 239 0.8744770 0.1255230  
## 246 0.2147651 0.7852349  
## 247 0.2147651 0.7852349  
## 249 0.2147651 0.7852349  
## 250 0.8744770 0.1255230  
## 255 0.8744770 0.1255230  
## 257 0.8744770 0.1255230  
## 266 0.2147651 0.7852349  
## 267 0.2147651 0.7852349  
## 268 0.8744770 0.1255230  
## 273 0.8744770 0.1255230  
## 274 0.8744770 0.1255230  
## 285 0.8744770 0.1255230  
## 292 0.8744770 0.1255230  
## 295 0.8744770 0.1255230  
## 300 0.8744770 0.1255230  
## 303 0.2147651 0.7852349  
## 305 0.8744770 0.1255230  
## 316 0.8744770 0.1255230  
## 319 0.8744770 0.1255230  
## 320 0.8744770 0.1255230  
## 321 0.8744770 0.1255230  
## 326 0.8744770 0.1255230  
## 328 0.2147651 0.7852349  
## 331 0.8744770 0.1255230  
## 335 0.8744770 0.1255230  
## 336 0.2147651 0.7852349  
## 342 0.8744770 0.1255230  
## 345 0.8744770 0.1255230  
## 347 0.8744770 0.1255230  
## 348 0.2147651 0.7852349  
## 354 0.8744770 0.1255230  
## 362 0.8744770 0.1255230  
## 377 0.8744770 0.1255230  
## 379 0.8744770 0.1255230  
## 380 0.2147651 0.7852349  
## 383 0.8744770 0.1255230  
## 384 0.8744770 0.1255230  
## 390 0.8744770 0.1255230  
## 394 0.8744770 0.1255230  
## 402 0.8744770 0.1255230  
## 406 0.8744770 0.1255230  
## 413 0.8744770 0.1255230  
## 416 0.8744770 0.1255230  
## 418 0.8744770 0.1255230  
## 422 0.2147651 0.7852349  
## 425 0.8744770 0.1255230  
## 427 0.8744770 0.1255230  
## 428 0.8744770 0.1255230  
## 431 0.8744770 0.1255230  
## 434 0.2147651 0.7852349  
## 436 0.8744770 0.1255230  
## 439 0.8744770 0.1255230  
## 445 0.8744770 0.1255230  
## 447 0.8744770 0.1255230  
## 448 0.8744770 0.1255230  
## 450 0.8744770 0.1255230  
## 457 0.2147651 0.7852349  
## 458 0.8744770 0.1255230  
## 462 0.8744770 0.1255230  
## 463 0.2147651 0.7852349  
## 467 0.8744770 0.1255230  
## 469 0.2147651 0.7852349  
## 470 0.8744770 0.1255230  
## 481 0.8744770 0.1255230  
## 488 0.8744770 0.1255230  
## 490 0.8744770 0.1255230  
## 500 0.8744770 0.1255230  
## 504 0.2147651 0.7852349  
## 506 0.2147651 0.7852349  
## 507 0.8744770 0.1255230  
## 511 0.8744770 0.1255230  
## 525 0.2147651 0.7852349  
## 529 0.8744770 0.1255230  
## 543 0.8744770 0.1255230  
## 546 0.8744770 0.1255230  
## 547 0.8744770 0.1255230  
## 549 0.8744770 0.1255230  
## 552 0.2147651 0.7852349  
## 568 0.2147651 0.7852349  
## 570 0.8744770 0.1255230  
## 573 0.8744770 0.1255230  
## 575 0.8744770 0.1255230  
## 581 0.2147651 0.7852349  
## 588 0.8744770 0.1255230  
## 591 0.8744770 0.1255230  
## 595 0.8744770 0.1255230  
## 607 0.8744770 0.1255230  
## 611 0.8744770 0.1255230  
## 620 0.8744770 0.1255230  
## 622 0.8744770 0.1255230  
## 623 0.2147651 0.7852349  
## 626 0.8744770 0.1255230  
## 632 0.2147651 0.7852349  
## 633 0.2147651 0.7852349  
## 635 0.2147651 0.7852349  
## 640 0.8744770 0.1255230  
## 642 0.8744770 0.1255230  
## 648 0.2147651 0.7852349  
## 651 0.2147651 0.7852349  
## 655 0.8744770 0.1255230  
## 659 0.8744770 0.1255230  
## 662 0.8744770 0.1255230  
## 675 0.8744770 0.1255230  
## 680 0.8744770 0.1255230  
## 683 0.8744770 0.1255230  
## 694 0.2147651 0.7852349  
## 696 0.2147651 0.7852349  
## 702 0.8744770 0.1255230  
## 706 0.8744770 0.1255230  
## 708 0.8744770 0.1255230  
## 710 0.2147651 0.7852349  
## 713 0.8744770 0.1255230  
## 716 0.8744770 0.1255230  
## 724 0.8744770 0.1255230  
## 725 0.2147651 0.7852349  
## 729 0.2147651 0.7852349  
## 732 0.8744770 0.1255230  
## 733 0.8744770 0.1255230  
## 745 0.8744770 0.1255230  
## 752 0.8744770 0.1255230  
## 753 0.8744770 0.1255230  
## 762 0.8744770 0.1255230  
## 766 0.8744770 0.1255230  
## 769 0.8744770 0.1255230  
## 772 0.2147651 0.7852349  
## 775 0.2147651 0.7852349  
## 780 0.2147651 0.7852349  
## 788 0.2147651 0.7852349  
## 789 0.8744770 0.1255230  
## 791 0.8744770 0.1255230  
## 797 0.8744770 0.1255230  
## 800 0.8744770 0.1255230  
## 801 0.2147651 0.7852349  
## 802 0.8744770 0.1255230  
## 813 0.2147651 0.7852349  
## 815 0.8744770 0.1255230  
## 816 0.8744770 0.1255230  
## 826 0.2147651 0.7852349  
## 831 0.8744770 0.1255230

ROC\_curve\_dt<-roc(test$HighAud, tree\_classTest[,"Yes"])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(ROC\_curve\_dt)



auc(ROC\_curve\_dt)

## Area under the curve: 0.8745

The graph shown above shows that the Area under the curve takes up 87.45% which means that the performance of the model has an average time distinguishing between the positive and negative classes. Overall, this model would be an okay graph to use. This graph will be saved in order to compare how well all the models interpret the test data.

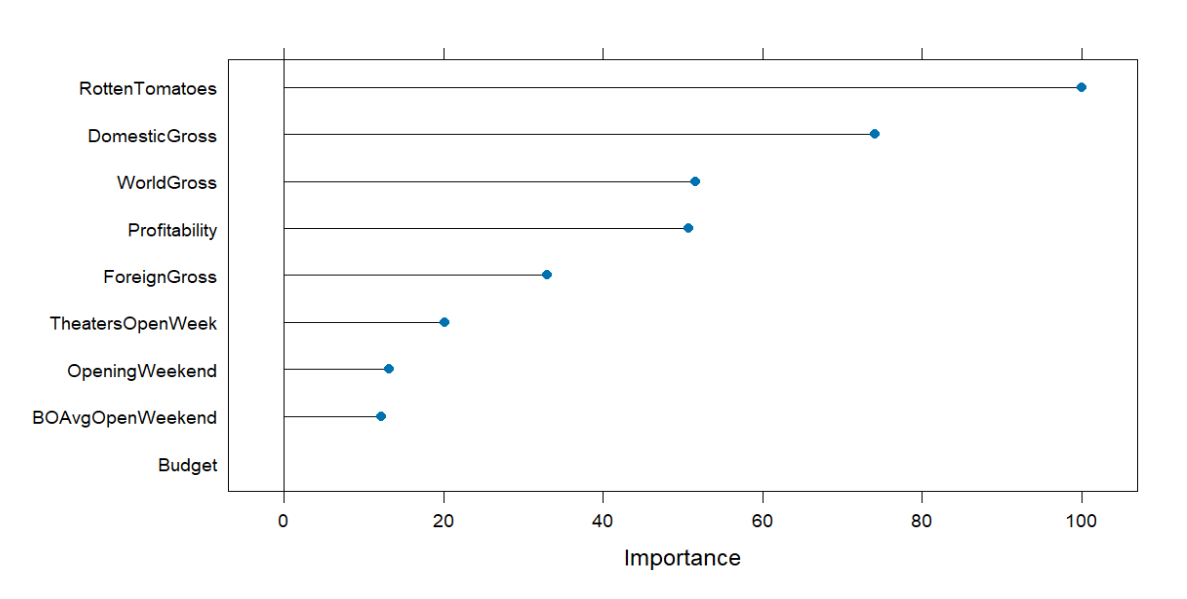
### Bagging

bag\_movie<-train(HighAud~.,data=train, method="treebag", trControl=cvcontrol, importance=TRUE)  
  
bag\_movie

## Bagged CART   
##   
## 627 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 1 times)   
## Summary of sample sizes: 564, 565, 564, 565, 564, 564, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8373016 0.5870431

The bagging will be used to improve accuracy in the machine learning algorithm. The bag\_movie contains 2 classes called Yes and No that indicate if the movie agrees with HighAud. It also has an optimal model with an accuracy of 0.8373016 and a kappa value of 0.5870431, which is inadequate since Accuracy and the Kappa are so low, which says that the overall agreement level is poor.

plot(varImp(bag\_movie))



The varImp plot shown above shows that RottenTomatoes, DomesticGross, and WorldGross are the most important to the bagged HighAud variable with importance values of 100, 75, and 55. This means that the project should consider these variables as important to the training and testing of the variable HighAud.

bagg\_pred<-predict(bag\_movie, newdata = test, type="raw")  
head(bagg\_pred)

## [1] Yes No No Yes No No   
## Levels: No Yes

confusionMatrix(test$HighAud, bagg\_pred)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 137 12  
## Yes 22 37  
##   
## Accuracy : 0.8365   
## 95% CI : (0.7791, 0.8841)  
## No Information Rate : 0.7644   
## P-Value [Acc > NIR] : 0.007172   
##   
## Kappa : 0.5761   
##   
## Mcnemar's Test P-Value : 0.122713   
##   
## Sensitivity : 0.8616   
## Specificity : 0.7551   
## Pos Pred Value : 0.9195   
## Neg Pred Value : 0.6271   
## Prevalence : 0.7644   
## Detection Rate : 0.6587   
## Detection Prevalence : 0.7163   
## Balanced Accuracy : 0.8084   
##   
## 'Positive' Class : No   
##

The accuracy of the confusion matrix and Statistics is relatively average since it is 0.8365. The 95% confidence interval is (0.7791, 0.8841). The kappa is relatively low and has a value of 0.5761, which means that there is a poor overall agreement level. This confusion matrix also claims that the p-value is 0.007172, which is very low and shows it greatly goes against the null hypothesis. However, this model is not a good choice since it has mediocre Accuracy and an extremely low Kappa.

bagg\_probs <- predict(bag\_movie, newdata = test, type = "prob")  
head(bagg\_probs)

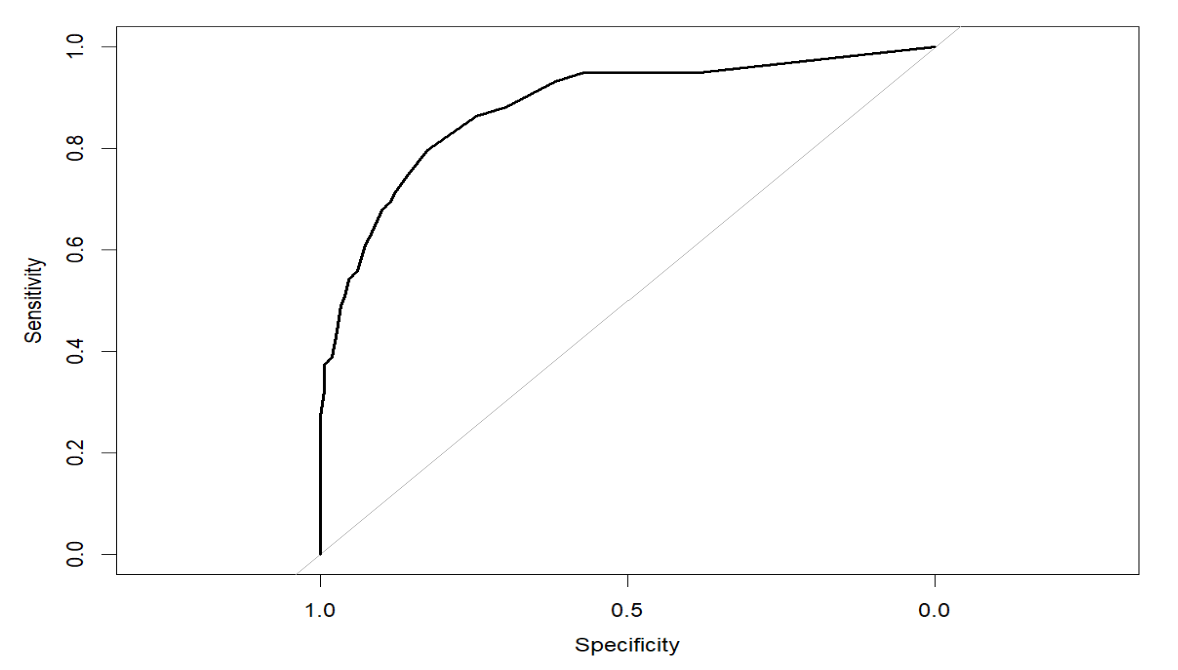
## No Yes  
## 1 0.64 0.36  
## 2 0.80 0.20  
## 3 0.60 0.40  
## 4 0.12 0.88  
## 5 0.00 1.00  
## 6 0.92 0.08

ROC\_curve\_bagg <- roc(test$HighAud, bagg\_probs[ , "Yes"])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(ROC\_curve\_bagg)



auc(ROC\_curve\_bagg)

## Area under the curve: 0.8832

The graph shown above shows that the Area under the curve takes up 88.32% which means that the performance of the model can distinguish between the positive and negative classes. This graph will be saved in order to compare how well all the models interpret the test data.

### Random Forest

rf\_HighAud<-train(HighAud~., data=train, method="rf", trControl=cvcontrol, importance =TRUE)  
rf\_HighAud

## Random Forest   
##   
## 627 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 1 times)   
## Summary of sample sizes: 564, 564, 565, 564, 564, 564, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.8612647 0.6393875  
## 5 0.8612903 0.6411876  
## 9 0.8533794 0.6183443  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 5.

The random forest method will be used to improve the accuracy of the algorithms used in the machine learning algorithm. It contains 2 classes called Yes and No that indicate if the movie agrees with HighAud, and used a resampling of 10-fold classification that is processed once. It also has an optimal model with an accuracy of 0.8612903, kappa value of 0.6411876, and a mtry of 5. This Random Forest is relatively average due to the mediocre Accuracy and extremely low Kappa value.

rf\_pred<-predict(rf\_HighAud, newdata=test, type="raw")  
confusionMatrix(test$HighAud, rf\_pred)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 135 11  
## Yes 17 42  
##   
## Accuracy : 0.8654   
## 95% CI : (0.8114, 0.9086)  
## No Information Rate : 0.7452   
## P-Value [Acc > NIR] : 1.691e-05   
##   
## Kappa : 0.6583   
##   
## Mcnemar's Test P-Value : 0.3447   
##   
## Sensitivity : 0.8903   
## Specificity : 0.7925   
## Pos Pred Value : 0.9262   
## Neg Pred Value : 0.7119   
## Prevalence : 0.7452   
## Detection Rate : 0.6635   
## Detection Prevalence : 0.7163   
## Balanced Accuracy : 0.8414   
##   
## 'Positive' Class : No   
##

The accuracy of the confusion matrix and Statistics is relatively average since it is 0.8654. The 95% confidence interval is (0.8114, 0.9086). The kappa is relatively average and has a value of 0.6583, which means that there is a poor average overall agreement level. This confusion matrix also claims that the p-value is 1.691e-05, which is very low, and it does go against the null hypothesis. However, this model is not a good choice since it has low Accuracy and low Kappa.

rf\_probs<-predict(rf\_HighAud, newdata = test, type = "prob")  
head(rf\_probs)

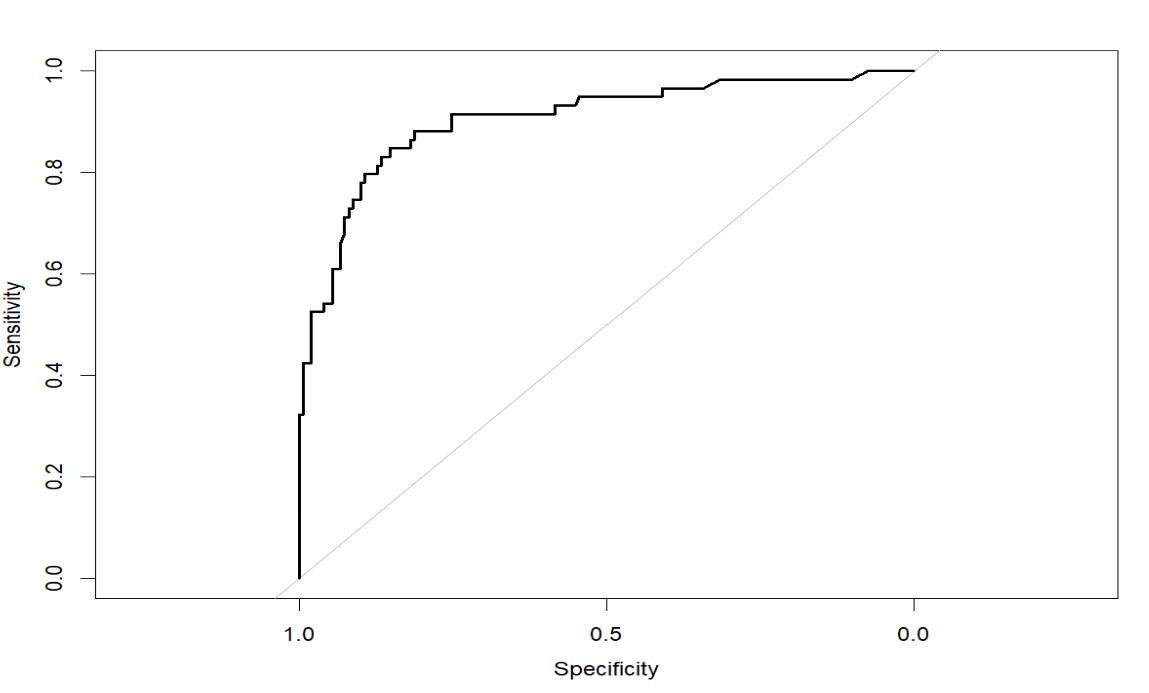
## No Yes  
## 1 0.408 0.592  
## 21 0.786 0.214  
## 23 0.996 0.004  
## 35 0.030 0.970  
## 38 0.930 0.070  
## 44 0.736 0.264

ROC\_curve\_rf<-roc(test$HighAud, rf\_probs[,"Yes"])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(ROC\_curve\_rf)



auc(ROC\_curve\_rf)

## Area under the curve: 0.9046

The graph shown above shows that the Area under the curve takes up 90.46% which means that the performance of the model can easily distinguish between the positive and negative classes. This graph will be saved in order to compare how well all the models interpret the test data.

### Boosting

gbm\_movie<-train(HighAud~., data=train,  
 method="gbm", verbose=F, trControl=cvcontrol)  
gbm\_movie

## Stochastic Gradient Boosting   
##   
## 627 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 1 times)   
## Summary of sample sizes: 564, 565, 564, 565, 565, 564, ...   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.8581157 0.6352836  
## 1 100 0.8565284 0.6319474  
## 1 150 0.8628520 0.6513151  
## 2 50 0.8660010 0.6585392  
## 2 100 0.8612647 0.6427977  
## 2 150 0.8596774 0.6430267  
## 3 50 0.8805428 0.6884288  
## 3 100 0.8676907 0.6615465  
## 3 150 0.8629544 0.6531492  
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
##   
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 50, interaction.depth =  
## 3, shrinkage = 0.1 and n.minobsinnode = 10.

The Stochastic Gradient Boosting with resampling of 10-fold cross-validation shows the most optimal model has 50 trees, an interactive depth of 3, shrinkage of 0.1, Accuracy of 0.8805428, and Kappa of 0.6884288, which can be said as a relatively average model because of its low Kappa despite having a decent Accuracy.

gbm\_preds <- predict(gbm\_movie, newdata = test, type = "raw")  
# get confusion matrix  
confusionMatrix(test$HighAud, gbm\_preds)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 138 11  
## Yes 24 35  
##   
## Accuracy : 0.8317   
## 95% CI : (0.7738, 0.8799)  
## No Information Rate : 0.7788   
## P-Value [Acc > NIR] : 0.03662   
##   
## Kappa : 0.5564   
##   
## Mcnemar's Test P-Value : 0.04252   
##   
## Sensitivity : 0.8519   
## Specificity : 0.7609   
## Pos Pred Value : 0.9262   
## Neg Pred Value : 0.5932   
## Prevalence : 0.7788   
## Detection Rate : 0.6635   
## Detection Prevalence : 0.7163   
## Balanced Accuracy : 0.8064   
##   
## 'Positive' Class : No   
##

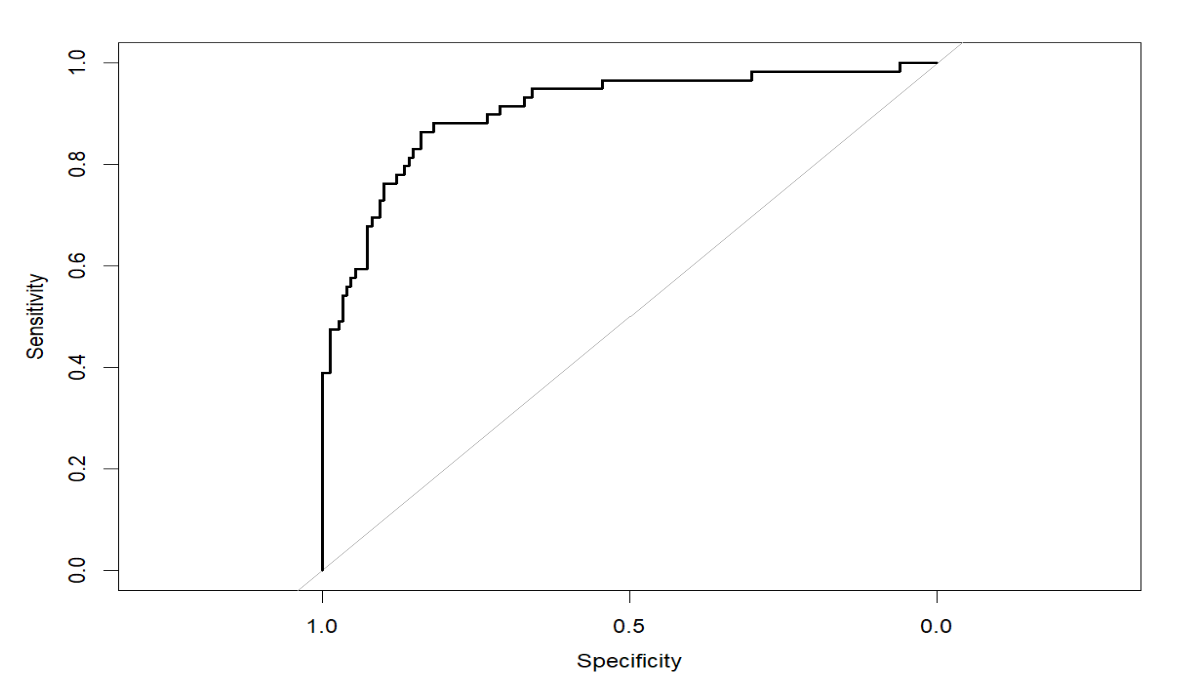
The confusionMatrix and Statistics for this section show the Accuracy is 0.8317, the 95% confidence interval has a range of (0.7738, 0.8799), the Kappa value of 0.5564, and the p-value of 0.03662, which means that it goes against the null hypothesis.

gbm\_probs<-predict(gbm\_movie, newdata = test, type = "prob")  
ROC\_curve\_gbm<-roc(test$HighAud, gbm\_probs[,"Yes"])

## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

plot(ROC\_curve\_gbm)



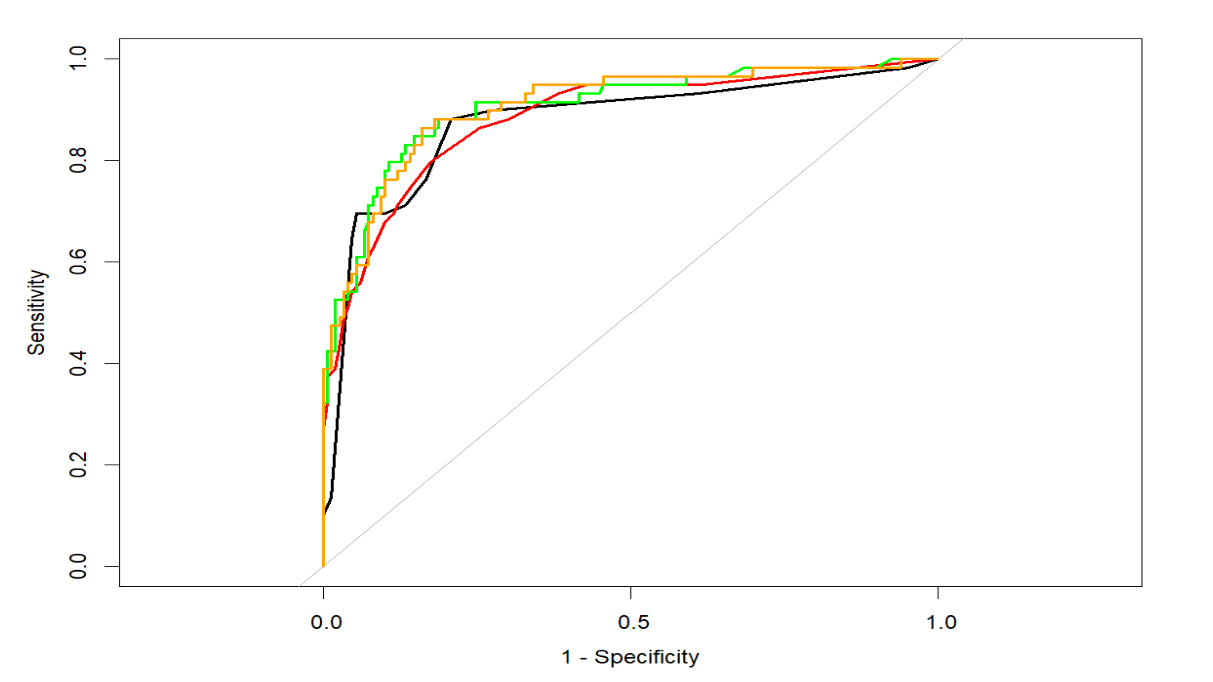
auc(ROC\_curve\_gbm)

## Area under the curve: 0.906

The graph shown above shows that the Area under the curve takes up 0.906, which means that the performance of the model can easily distinguish between the positive and negative classes. This graph will be saved in order to compare how well all the models interpret the test data. Stochastic Gradient Boosting shows better performance compared to the other models.

### Model Comparison High AudienceScore

# using the `add = TRUE` option for each plot after the first one:  
plot(ROC\_curve\_dt, col = "black", legacy.axes = T, ) # black  
plot(ROC\_curve\_bagg, add = TRUE, col = "red") # color red for bagging  
plot(ROC\_curve\_rf, add = TRUE, col = "green") # color green is for random forest  
plot(ROC\_curve\_gbm,add = TRUE, col = "orange") # color orange is for boosting



auc(ROC\_curve\_dt)

## Area under the curve: 0.8745

auc(ROC\_curve\_bagg)

## Area under the curve: 0.8832

auc(ROC\_curve\_rf)

## Area under the curve: 0.9046

auc(ROC\_curve\_gbm)

## Area under the curve: 0.906

The graphs shown above show the AUC-ROC of the decision tree, bagging, random forest, and Boosting. The best method for the variable, HighAud, is Boosting which is shown on the orange graph since it has the highest AUC-ROC by a small margin.

# Conclusion

One of the biggest global markets in the entertainment industry is the box office industry. According to IMDbPro, the industry has a yearly average box office revenue of around 11 to 14 million dollars and a yearly total gross of around 9 to 12 billion dollars. The project wanted to determine what makes a Hollywood movie have a high Profitability what makes it have a high AudienceScore, and how Profitability and AudienceScore correlate with each other. This was done with EDA, clustering, decision trees, bagging, boosting, and random forests that will help determine the pattern for high-earning and highly-rated movies.

The Profitability vs. AudienceScore Grouped by HighProfitsection shows a model of the relationship between Profitability and AudienceScore that is clustered with the variable, HighProfit, which determines if a movie is highly-profitable or not. The ScatterPlot of AudienceScore vs Profitability with Linear Method for lowly profited movies has a slightly, positive slope, while highly-profited movies have also a negative slope. The ScatterPlot of AudienceScore vs Profitability with Linear Method without Outliers for lowly profited movies have a positive slope, while highly-profited movies have also a positive slope. These scatterplots show that as more highly-profiting outliers appear in the graph, the slope of lowly-profiting movies and the slope of highly-profiting movies decreases, which could mean that highly-profiting outlier has a negative correlation to the slope of lowly-profiting movies and the slope of highly-profiting movies.

The Profitability and AudienceScore K-Means Clustering uses the elbow method and K-means clustering for AudienceScore. The project chose the k values 2, 3, and 4 for K-means clustering since they are the most optimal k-values based on the elbow method and the silhouette method. The clusters in all k-means clustering are overlapped with the other clusters within their graphs. Based on this, Hollywood Movie Cluster Plot with k=3 and Hollywood Movie Cluster Plot with k=2 have better clusters than Hollywood Movie Cluster Plot with k=4 since there is too much overlap to analyze. Hollywood Movie Cluster Plot with k=3 would be the most optimal since it has more than 2 variables and has less overlap than Hollywood Movie Cluster Plot with k=4.

The model comparison of the Profitibilitycolumn includes decision trees (with and without categorical variables), bagging, random forest, and boosting. The AUC-ROC curve shows that the best method for the variable, HighAud, is Stochastic Gradient Boosting since it has the highest AUC-ROC by a small margin.

The model comparison of the AudienceScorecolumn includes decision trees (with and without categorical variables), bagging, random forest, and boosting. The best method for the variable, HighAud, is the Stochastic Gradient Boosting since it has the highest AUC-ROC by a small margin.

This project taught me that usually low Kappa has an average or low Accuracy. It also taught me that the rpart() and rpart.plot() makes unique decision tree in every re-run, which was difficult for me to deal with because I had to constantly change the text to correlate to the decision trees.

Movie studios would find this information useful since it could help them determine the best kind of movies that will lead to larger profits and more positive feedback on their movies and firms for better publicity and recognition.

# Resources

Fonseca, L. (2019, August 15). Clustering analysis in R using K-means. Retrieved April 24, 2021, from <https://towardsdatascience.com/clustering-analysis-in-r-using-k-means-73eca4fb7967>

Hidayatuloh, A. (n.d.). Visualize Clustering Using ggplot2. Retrieved April 24, 2021, from <https://rpubs.com/aephidayatuloh/clustervisual>

Michel Kana, P. (2020, February 24). Data clustering tutorial for advanced. Retrieved April 24, 2021, from <https://towardsdatascience.com/clustering-for-data-nerds-ebbfb7ed4090>

K-means cluster analysis. (n.d.). Retrieved April 24, 2021, from <https://afit-r.github.io/kmeans_clustering>

Rashmi, Kassambara 06 May 2020 The demo data used in this tutorial is available in the default installation of R. Juste type data(“USArrests”) Reply, & Kassambara. (2018, October 21). K-Means clustering in R: Algorithm and practical examples. Retrieved April 24, 2021, from <https://www.datanovia.com/en/lessons/k-means-clustering-in-r-algorith-and-practical-examples/>