TEAM 3 | 3-19-19 | IST707 | GROUP PROJECT

INTERNET MOVIE DATABASE

Ali Ho

Kendra Osburn

James Robertson

Introduction

43 billion dollars. This is the revenue of the film industry in the US in 2018[1]. This is a competitive industry with a large chunk of money that is up for grabs. It has an average growth of 2% each year which makes the need for cutting edge analytics even more urgent. For too long data has been put in the corner. Like Jennifer Grey it needs to be front and center. Disruption is coming to the status quo and we're gonna make an offer you can't refuse.

Can we recommend movies? Recommending movies is a huge money maker. Netflix has run contests where they have paid out significant prizes for improvement to their recommendation algorithm. An additional application of this is in the realm of targeted marketing. Being able to recommend movies can be used to deliver movie advertisements to target audiences. This can also be used to produce advertisements that will target different demographics. Consider a movie starring George Clooney and Susan Sarandon. Audiences that like O Brother Where Art Thou and the Oceans trilogy could be delivered advertisements featuring George Clooney. While audiences that like Rocky Horror Picture Show and Thelma and Louise could be targeted by ads focussing on Susan Sarandon.

Where should the budget be spent to get desired results? Is the goal to maximize profit or optimize review scores? Many indie studios and aspiring directors have a strong desire for both. There are some ways for these directors get their feet wet. For example, Stephen King allows new directors to adapt any of his short stories for very little money. What other doors are available for these newcomers? Are there genres that are better for ratings and recognition? What about for profit or continued sustainability?

For larger production companies the focus shifts. These companies focus on having a few blockbuster movies referred to as "Tent Pole" movies. These "tent poles" provide buffers to the studios to produce riskier movies that might not be profitable but allow them to continue operating. There is a large risk to studios and their bottom line when one of these movies flops. Can flops be predicted from early ratings? If they can be predicted can the losses be minimized by working with theaters or a third party movie subscription service?

Analysis and Models

About the Data

The original data set contained 4,638 movies from multiple countries. The dataset was reduced to include only the movies that were produced by American production companies. The reduced dataset set contained 3,726 entries. The dataset was further reduced to only include stars who appeared in at least 5 movies. This yielded a dataset with 1,998 entries. The final reduction was by the most prolific directors, which resulted in a dataset containing 989 movies with 14 variables. Nas were checked for with sum(is.na), no nas were detected.

Variable	Data Type	Description
budget	Numeric	States the total budget for the movie
director	Factor	States the main director for the movie
genre	Factor	States the genre of the movie
gross	Numeric	States the total gross for the movie
name	Factor	States the title of the movie
rating	Ordered factor	States the rating of the movie: G, PG, PG-13 or R
released	Ordered factor	States the month that the movie was released
runtime	Numeric	States the total runtime of the movie

score	Numeric	States the score the movie received on IMDB
star	Factor	States the main star of the movie
votes	Numeric	States the number of votes the movie received on IMDB
writer	Factor	States the main writer for the movie
year	Ordered factor	States the year the movie was released
VARIABLES THAT WERE ADDED TO THE DATASET		
profit	Numeric	States the profit. Calculated from subtracting the budget from gross for each movie.
percProf	Numeric	States the percent profit. Calculated from dividing the profit by the budget for each movie.
starGender	Factor	States the gender of the star
directorGender	Factor	States the gender of the director
starAge	Numeric	States the age of the star when in the movie
directorAge	Numeric	States the age of the director when directing the movie
starPopularity	Numeric	States the star's popularity on TMBD
directorPopularity	Numeric	States the director's popularity on TMBD

Figure 1. Breakdown of the Dataset

A discretized dataset was created by quartiling each numeric column. The bottom 25% was discretized as "low", the middle 50% as "average" and the top 25% as "high". The only exception was for percent profit, where the data was discretized as "negative", "average" and then the top 25% as "high".

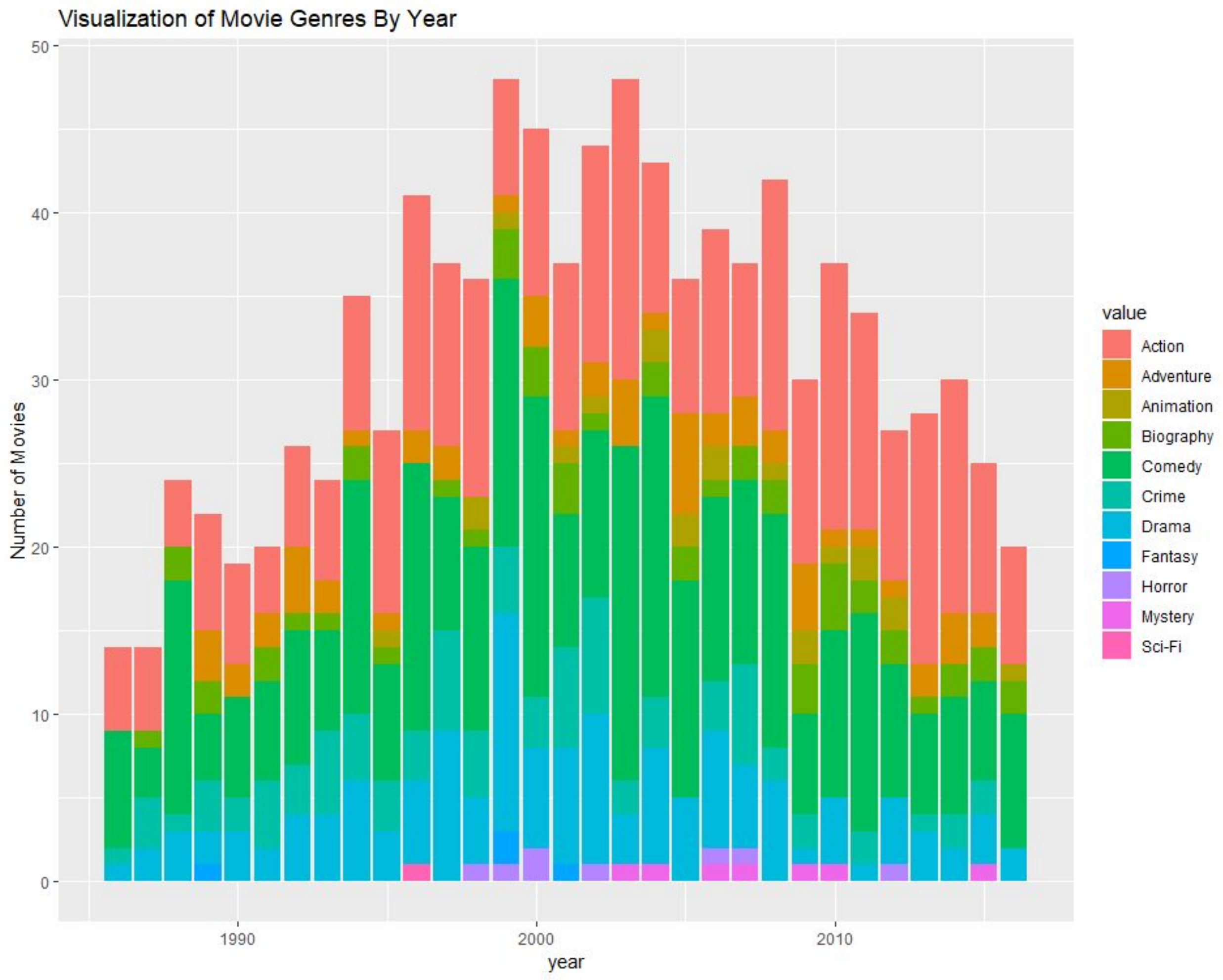
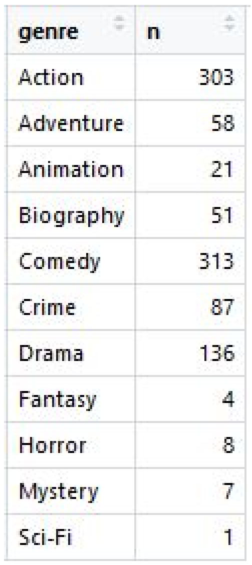


Figure 2. Visualization of Movies Genre By Year



genre	n
Action	303
Adventure	58
Animation	21
Biography	51
Comedy	313
Crime	87
Drama	136
Fantasy	4
Horror	8
Mystery	7
Sci-Fi	1

The distribution of movies by year appears to be normally distributed. Action, comedy and drama account for the majority of the movies in the dataset. The dataset contains only one science fiction movie, four fantasy movies, seven mysteries, and eight horror movies. To classify by genre all genres with the exception of action, comedy, and drama will be grouped in a category called other.

Figure 2a. Counts of genres

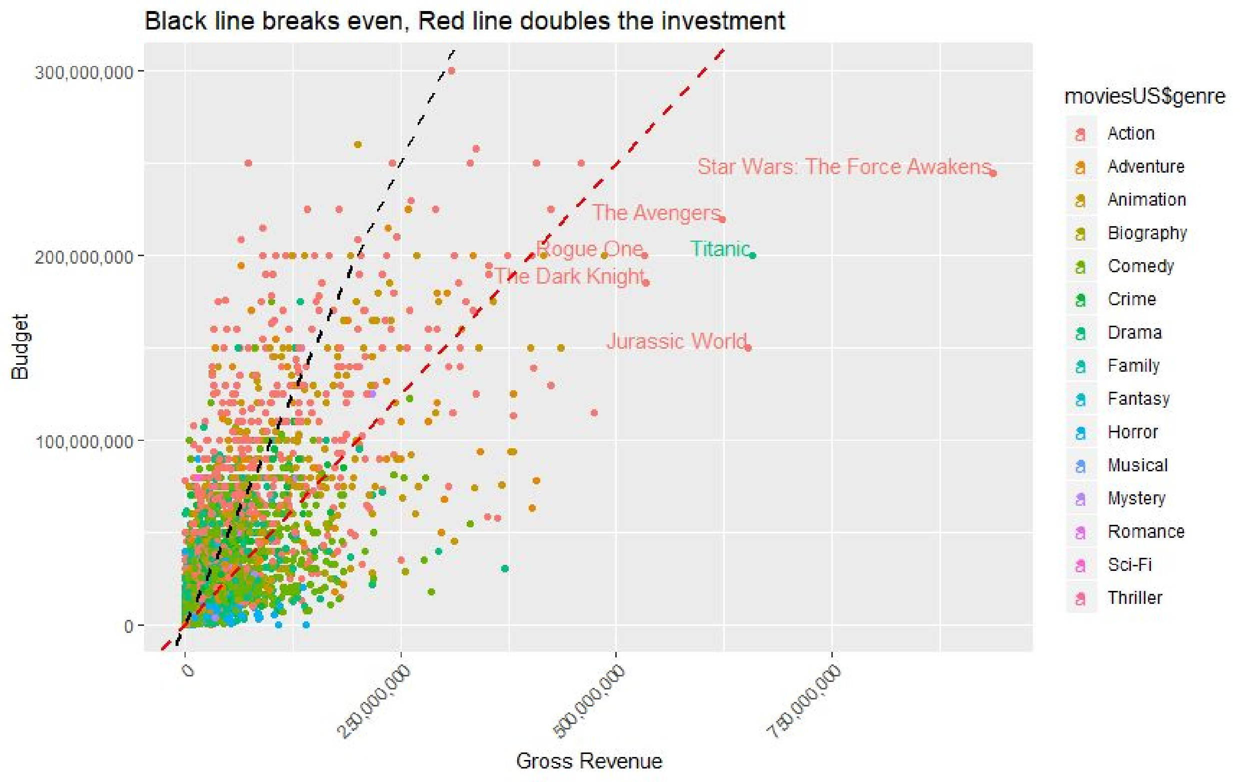


Figure 3. Visualization of Movies by Gross Revenue, Budget and Genre

Titanic is the only movie, not in the action genre, to be in the top 6 movies for gross. Movies to the left of the black line lose money, movies to the right of the line break even to double their investment and movies to the right of the red line more than double their investment. The majority of the movies appear to have a budget of less than $100,000,000.

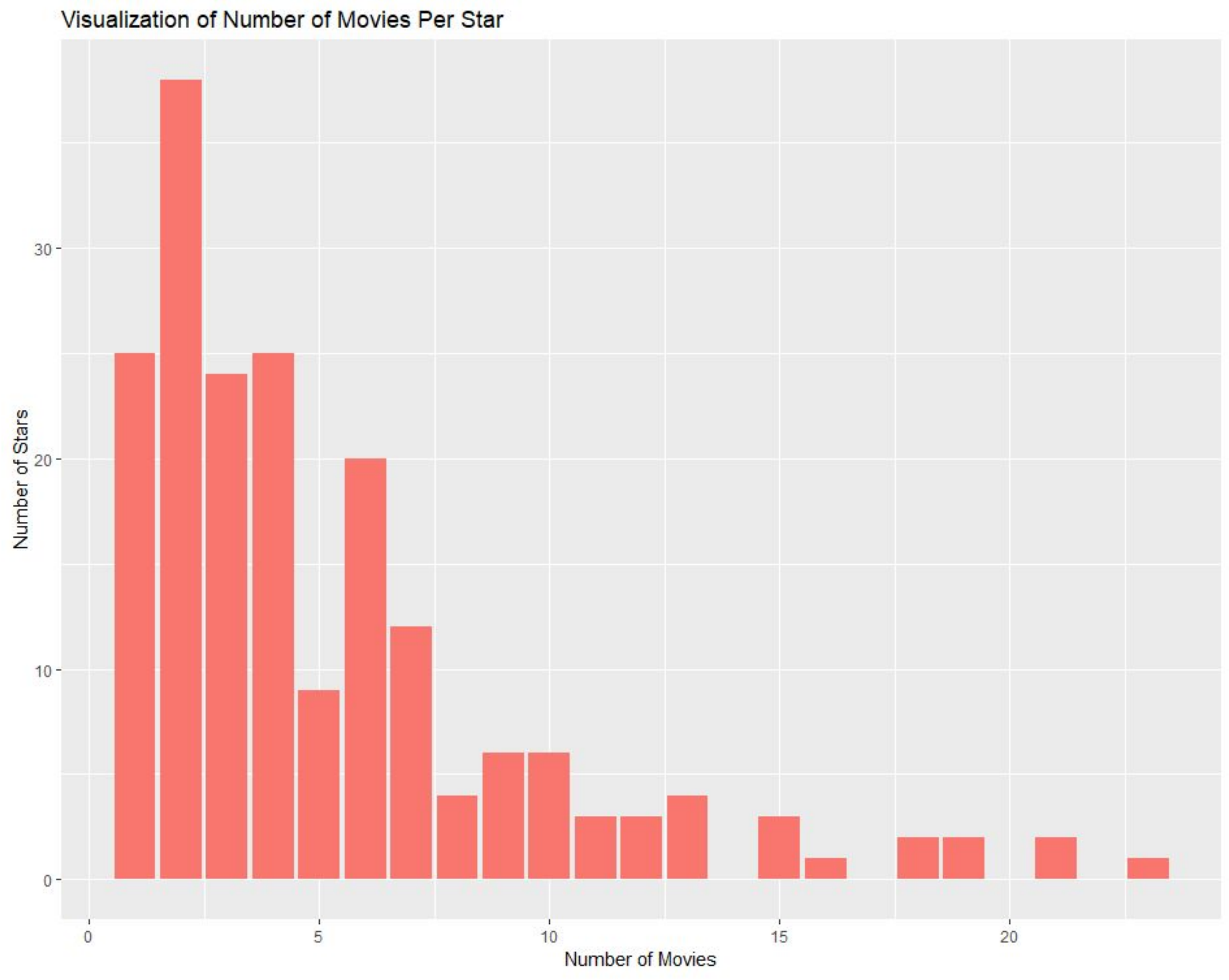


Figure 4. Visualization of Number of Movies per Star

The majority of the 190 stars in the dataset appear in less than seven movies. There are 21 stars who appear in more than ten movies. The most prolific stars are Tom Hanks, Nicolas Cage and Tom Cruise who star in more than 20 movies in the dataset.

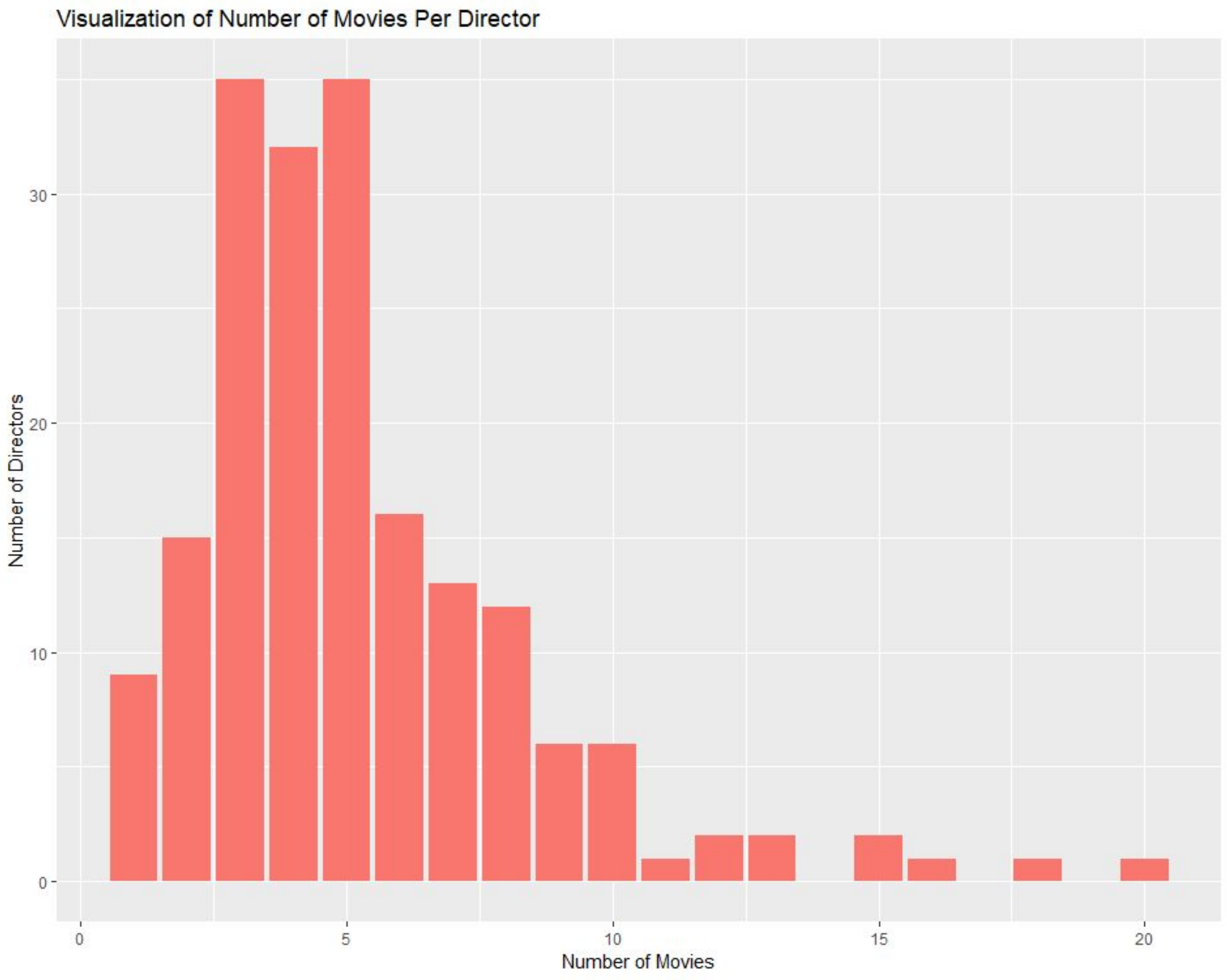


Figure 5. Visualization of Number of Movies per Director

The dataset contains 189 directors. The majority of the directors directed 3 to 5 movies within the dataset. The most prolific directors are Clint Eastwood, Ron Howard and Woody Allen who all directed more than 15 movies in the dataset.

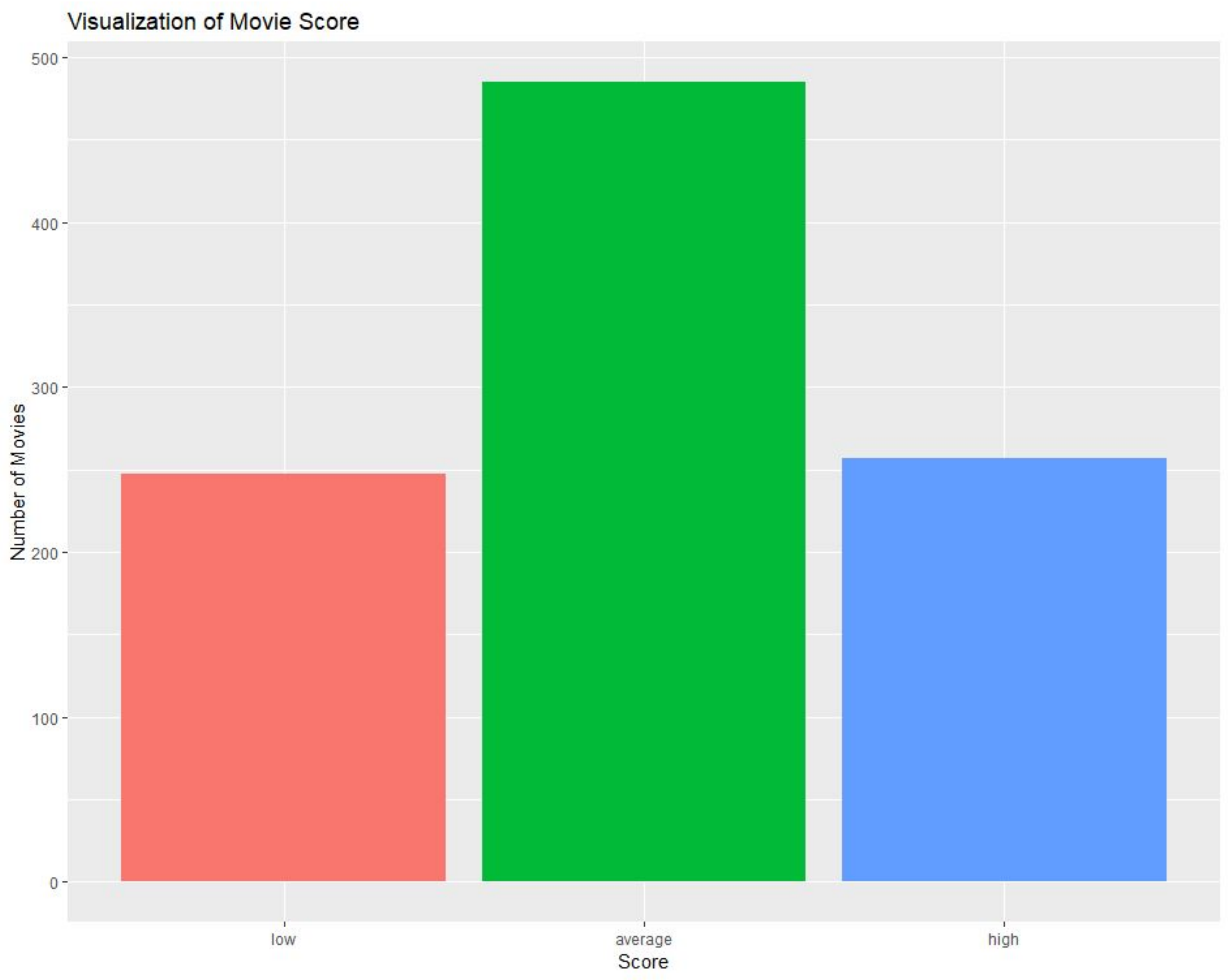


Figure 6. Visualization of Movie Score Discretized

The score were descritized so that scores of 6.1 or lower were classified as low, scores between 6.1 and 7.2 as average and scores of 7.2 or higher as high. This method resulted in 25% of the movies classified as low, 50% of the movies as average and 25% as high.

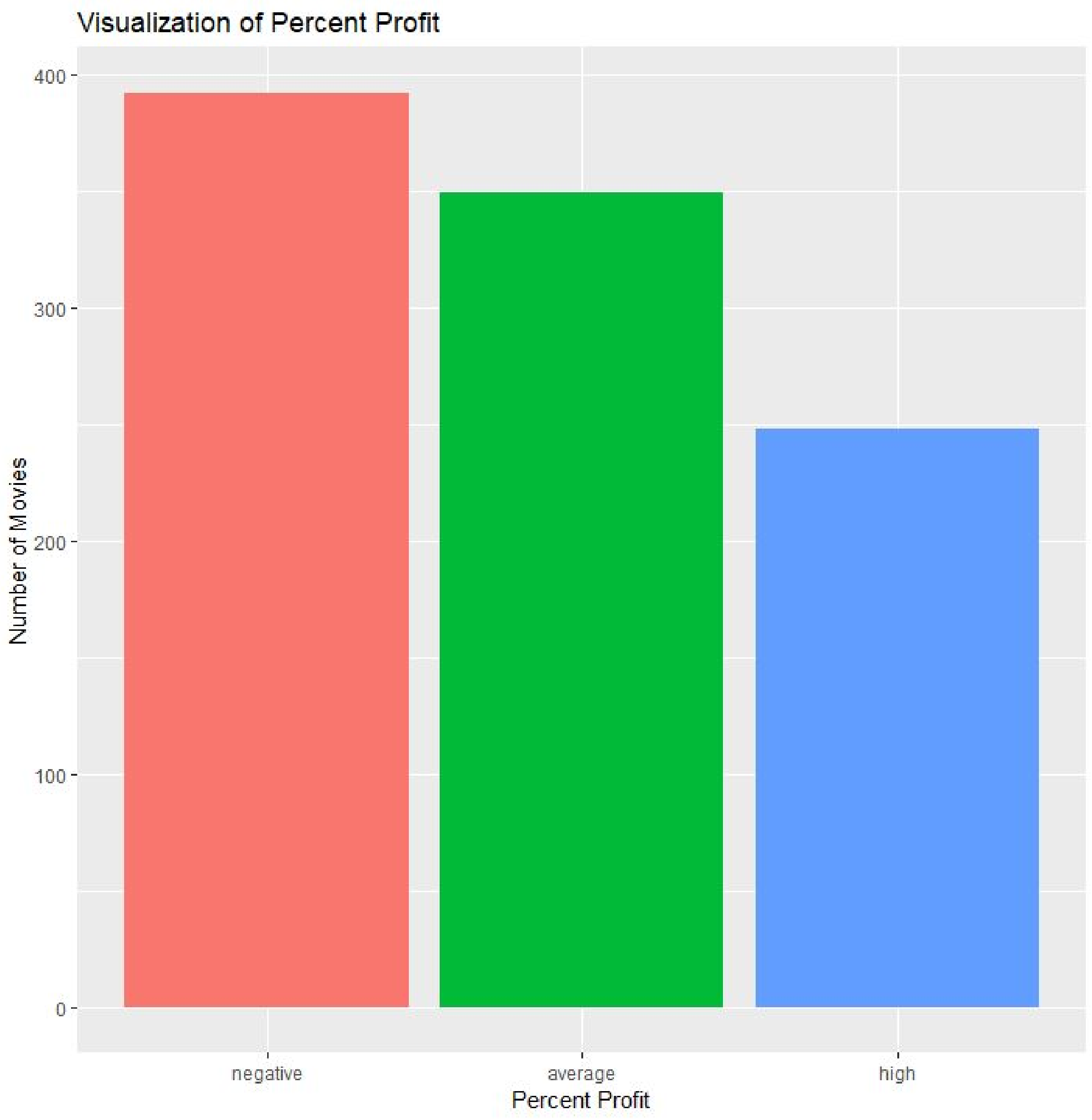


Figure 7. Visualization of Percent Profit Discretized

Percent profit was descritized into three categories: negative, average, and high. Negative accounts for all movies that made a profit less than their budget, average is described as a percent profit greater than 0 but less than 1.17, and high is a percent profit 1.17 or greater. This method resulted in 392 movies with a negative percent profit, 349 movies with an average percent profit, and 248 movies with a high percent profit.

TEXT DATA

The researchers gathered the textual data by scraping IMDb for reviews. While most movies offered multiple reviews, the researchers chose to use a single review for each. This single review was chosen by the ranking algorithm that determines which review to display first on each IMDb movie page. Despite their best efforts, the researchers were unable to determine the specific criteria IMDb's algorithm used. (Was it the number of "helpful" votes? The number of past reviews authored by a given reviewer? Some combination of proprietary metrics hidden from IMDb users?). In any case, the researchers chose the algorithm's single review to represent each movie for reasons of project scope more than accuracy. While this methodology may serve the researchers in an educational context, it is clear that additional reviews would be necessary in order to make any statistically useful inferences.

SCRAPED IMDB & TMBD DATA

"A census taker once tried to test me. I ate his liver with some fava beans and a nice Chianti."

-IMDB and TMBD data, probably

One of the many offshoot projects that sprung up and out of the initial endeavor included a revised and updated scraper. This scraper combed not only IMDB (Internet Movie Database) but TMDB (The Movie Database) for in-depth actor/director/production information. This scraper, written in python with the aid of the Beautiful Soup Library, collected everything from actor and director gender to film trivia. Unfortunately only a small subset of this additional scraped data was able to be utilized by the researchers due to time constraints.

Models

ASSOCIATION RULE MINING

The purpose of association rule mining is to find associations between the variables. Through association rule mining, patterns emerge in the data that explain customers who purchase item A usually also purchase item C. Actionable intel is generated by utilizing the association rule mining technique. The function used for association rule mining is apriori which is from the arules library. The apriori function requires a discretized data frame, and the following parameters were used supp and conf.

$$\frac{amount\ of\ times\ items\ on\ the\ lhs\ and\ rhs\ appear\ together}{total\ amount\ of\ entries\ in\ the\ data\ set}$$

Figure 8. Formula for Support

Support is the amount of times that the items appear together in the data set divided by the total amount of entries in the dataset.

$$\frac{amount\ of\ times\ items\ on\ the\ lhs\ and\ rhs\ appear\ together}{total\ amount\ of\ times\ item(s) on\ the\ lhs\ appear\ in\ the\ data\ set}$$

Figure 9. Formula for Confidence

Confidence is the amount of times the items on the left-hand side and right-hand side appear together divided by the count of times the item(s) on the left-hand side appear in total.

$$\frac{\left(\frac{Amount\ of\ times\ items\ on\ the\ lhs\ and\ rhs\ appear\ together}{total\ amount\ of\ entries\ in\ the\ data\ set}\right)}{\left(\frac{amount\ of\ times\ item(s) on\ the\ rhs\ appear}{total\ amount\ of\ entries\ in\ the\ data\ set}\times\frac{amount\ of\ times\ item(s) on\ the\ lhs\ appear}{total\ amount\ of\ entries\ in\ the\ data\ set}\right)}$$

Figure 10. Formula for Lift

Lift is calculated by the count of amount of times that the items on the left-hand side and right-hand side appear together divided by the count of times the item(s) on the left-hand side appear multiplied by the count of times the item(s) on the right-hand side appear. Once the rules are generated and stored in a vector, it is possible to sort the rules by support, confidence or support. To do this use the sort() function. Finally, the inspect() function displays the list of the rules that were created.

K-MEANS CLUSTERING & HCLUST

k-means()

K-means is an unsupervised machine learning clustering algorithm. It is unsupervised because the labels are removed from the dataset. It requires all attributes to be numeric. K-means requires the data frame or distance matrix and the parameter for the number of centroids to generate (k). Nstart =50 was used to generate 50 initial groups of k clusters and then kmeans uses the best random generation of k clusters for the algorithm. Set.seed() was used to replicate the results. Once the random centroids are selected, kmeans starts an iterative process of recalculating the centroid every time a new entry is included in the cluster. Kmeans groups items based on similarity.

hclust()

hclust() is an unsupervised machine learning hierarchical clustering algorithm. All the attributes must be numeric. Hclust requires the data frame or distance matrix and takes the method parameter. The default method is complete. Hclust creates a dendrogram and the cutree() function is used to create a specific number of clusters.

DECISION TREES

rpart()

Rpart is a supervised machine learning recursive partitioning and regression tree technique. It is supervised because the training data contains the labels that are to be predicted. It requires all variables to be either numeric, discretized, or in bins. The target variable must be a factor. Rpart requires a formula, the data frame that is being used and the method. The method for classification tasks is “class”.

Optional parameters include control=rpart.control and parms = list(split = 'information'). Rpart() by default splits by gini. To split by information gain parms = list(split = 'information') was used. Rpart creates a decision tree model to predict the classification of entries in the data frame. Rpart.plot generate a visual of the decision tree with the root node, internal nodes and the leaf nodes.

NAIVE BAYES

naive_bayes()

Naïve Bayes is a supervised machine learning classification technique. The algorithm calculates the probability of the event happening. There are multiple libraries that can be used for the naïve bayes technique. The two libraries used in the analysis are naiveBayes and e1071. The models have the same requirements; however, the plotting options are different. The target variable must be a factor. Naïve bayes requires a formula and the data frame that is being used.

KNN

kNN()

kNN is a supervised machine learning classification algorithm. The algorithm uses the nearest neighbors (the closest points) to determine which class the entry belongs in. The training and test datasets must have the label removed, however the training label is used in the algorithm. kNN is part of the class library. It requires the training data frame without the classifying label, the test data frame, cl the classifications of the training dataset, k the number of nearest neighbors and prob equals true or false. The training labels must be a factor.

SVM

svm()

Support vector machine is a supervised machine learning technique mainly used for classification problems. The training data contains the labels that are to be predicted. Svm can transform the data into n-dimensional space and finds the hyperplane that best separates the data. The best hyperplane will have the maximum distance between classes. The optimal hyperplane will have a large margin between the support vectors (the data points for the different classes that are closest to the margin). Svm requires a formula, the data frame that is being used, the type of kernel, cost and whether the data needs to be scaled or not. There are three main kernels used for svm transformations: radial, polynomial and linear. A kernel transforms data into a new dimension so that a margin can be determined. Cost specifies the level of penalization for having points inside the hyperplane margin. If there is a low cost, then the points within the margins have less of a penalty than if there is a high cost. If scale equals TRUE, then the svm algorithm will automatically scale the data. However, if the data has already been scaled then it is necessary to put scale = FALSE. The svm algorithm is then used to predict the class of the test data.

RANDOM FOREST

randomForest()

Random forest is an ensembling supervised machine learning technique that trains multiple decision trees and then combines their results into one final model. Random forest can also be run as an unsupervised technique. It can be utilized in classification and regression problems. The random forest algorithm requires a formula and the training data frame. There are numerous parameters that can be implemented. One parameter is ntree which states the number of trees that are to be created. It is important that ntree is large enough to ensure that every row in the dataset receives multiple predictions.

TEXT-MINING

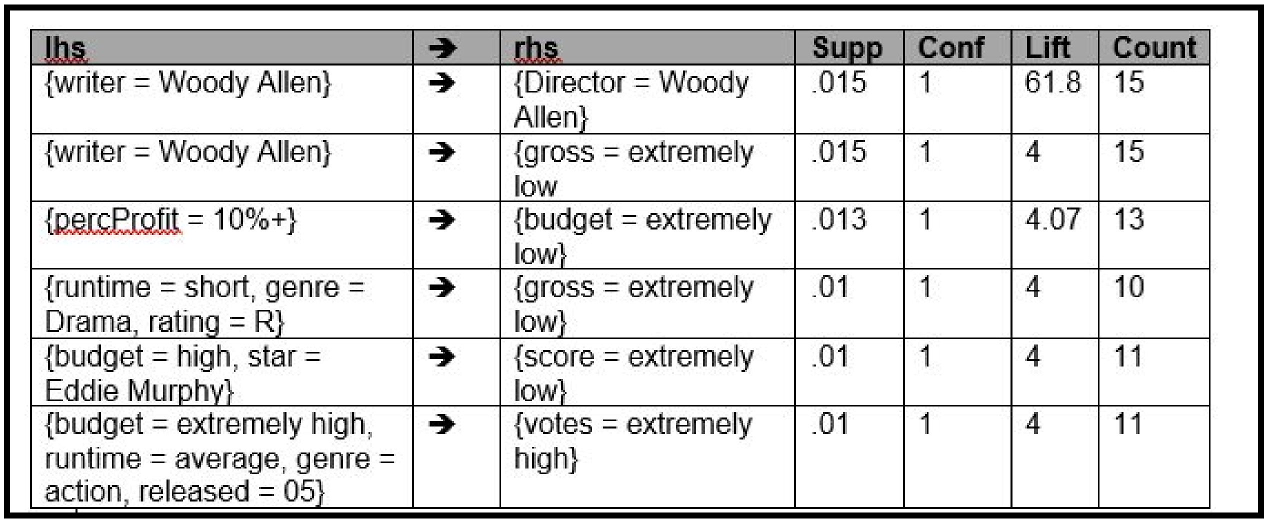
Text mining enables researchers to turn paragraphs of text into meaningful data by extracting individual words. Unlike many other models, text mining isn’t a singular method that can be applied to data. Rather, it is a collection of many different methods — including sentiment analysis, topic modeling, and lie detection — of transforming blocks of text into data that can be processed, classified, and analyzed. As such, there are many different libraries that can help achieve this textual information retrieval. The input is any block of text, while the output is some variation of record data, often in the form of a term document matrix.

Results

I. ASSOCIATION RULE MINING

THIS IS WHERE ALL THE FUN HAPPENED!! The researchers could have spent their entire project JUST focusing on association rule mining and tuning different parameters and scraping different sites. In fact, one researcher scraped every single cast member from every single movie in the dataset. Right down to makeup assistant #4. Unfortunately, due to limitations of time and energy, the researcher was unable to actually make use of this 9mb file.

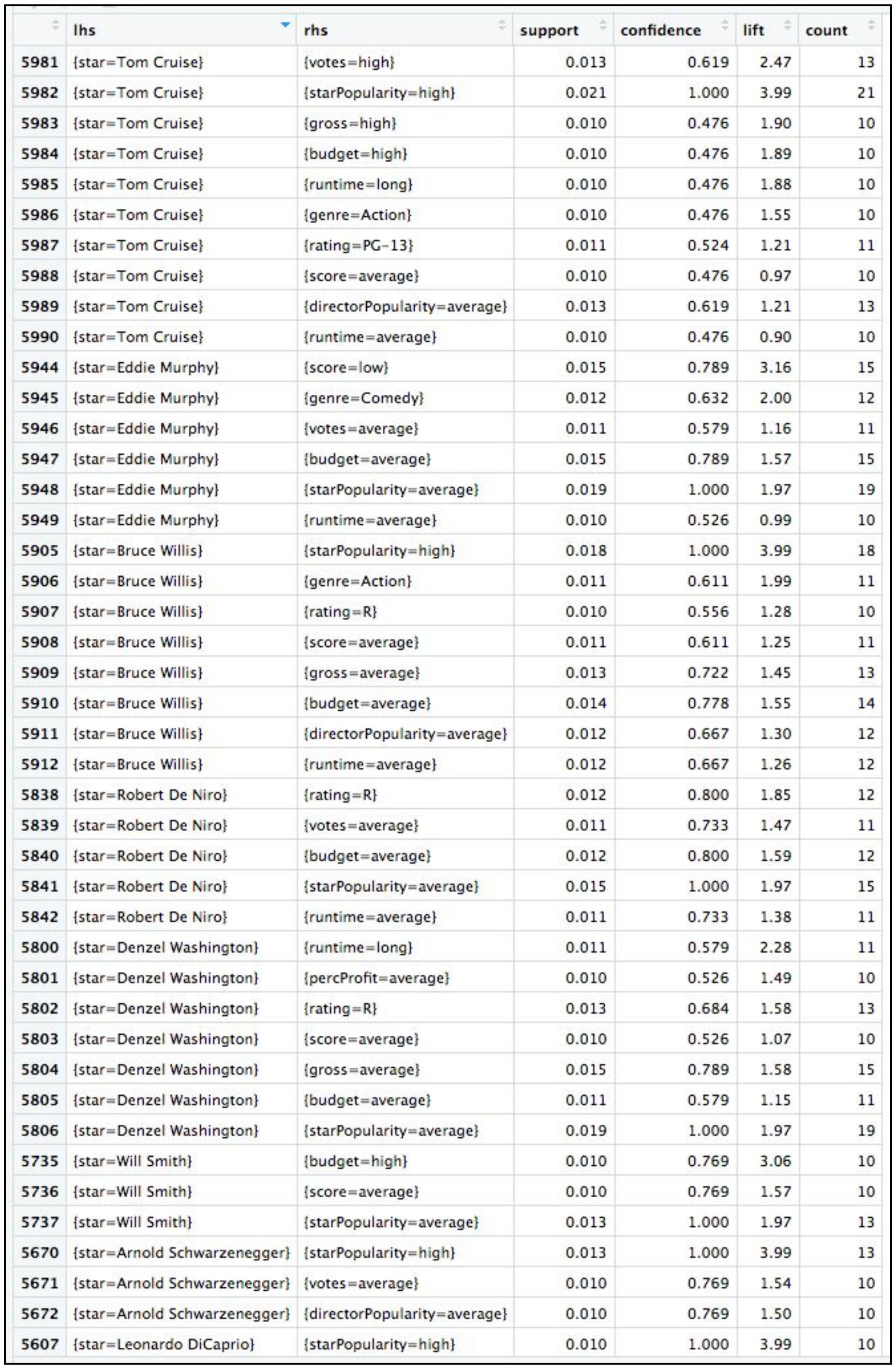
The data used in this model was descritized. In order to maximize the number of “best” rules, a loop was created to cycle through each attribute and put that attribute on the right hand side and funnel the results into a new data frame. From there, the data could be sorted more easily.



lhs	➔	rhs	Supp	Conf	Lift	Count
{writer = Woody Allen}	➔	{Director = Woody Allen}	.015	1	61.8	15
{writer = Woody Allen}	➔	{gross = extremely low	.015	1	4	15
{percProfit = 10%+}	➔	{budget = extremely low}	.013	1	4.07	13
{runtime = short, genre = Drama, rating = R}	➔	{gross = extremely low}	.01	1	4	10
{budget = high, star = Eddie Murphy}	➔	{score = extremely low}	.01	1	4	11
{budget = extremely high, runtime = average, genre = action, released = 05}	➔	{votes = extremely high}	.01	1	4	11

Figure 11.

STARS ON THE LHS, BY STAR

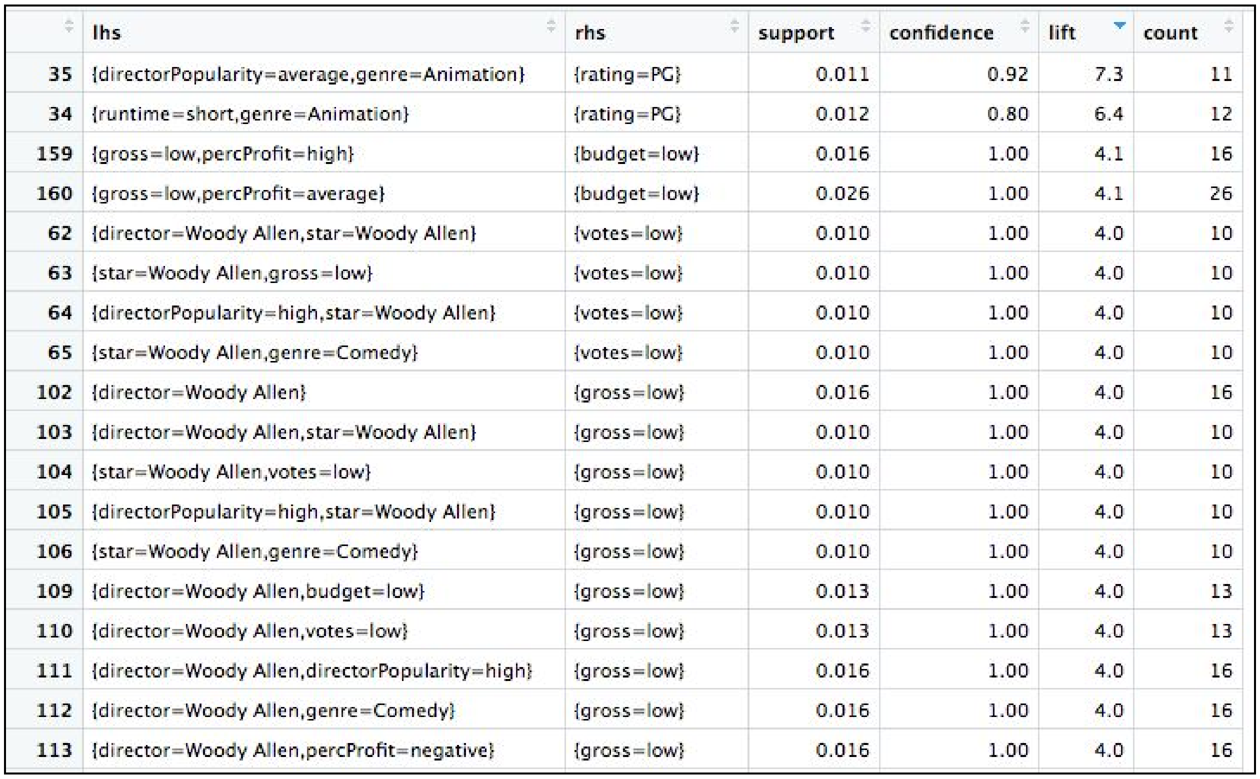


	lhs	rhs	support	confidence	lift	count
5981	{star=Tom Cruise}	{votes=high}	0.013	0.619	2.47	13
5982	{star=Tom Cruise}	{starPopularity=high}	0.021	1.000	3.99	21
5983	{star=Tom Cruise}	{gross=high}	0.010	0.476	1.90	10
5984	{star=Tom Cruise}	{budget=high}	0.010	0.476	1.89	10
5985	{star=Tom Cruise}	{runtime=long}	0.010	0.476	1.88	10
5986	{star=Tom Cruise}	{genre=Action}	0.010	0.476	1.55	10
5987	{star=Tom Cruise}	{rating=PG-13}	0.011	0.524	1.21	11
5988	{star=Tom Cruise}	{score=average}	0.010	0.476	0.97	10
5989	{star=Tom Cruise}	{directorPopularity=average}	0.013	0.619	1.21	13
5990	{star=Tom Cruise}	{runtime=average}	0.010	0.476	0.90	10
5944	{star=Eddie Murphy}	{score=low}	0.015	0.789	3.16	15
5945	{star=Eddie Murphy}	{genre=Comedy}	0.012	0.632	2.00	12
5946	{star=Eddie Murphy}	{votes=average}	0.011	0.579	1.16	11
5947	{star=Eddie Murphy}	{budget=average}	0.015	0.789	1.57	15
5948	{star=Eddie Murphy}	{starPopularity=average}	0.019	1.000	1.97	19
5949	{star=Eddie Murphy}	{runtime=average}	0.010	0.526	0.99	10
5905	{star=Bruce Willis}	{starPopularity=high}	0.018	1.000	3.99	18
5906	{star=Bruce Willis}	{genre=Action}	0.011	0.611	1.99	11
5907	{star=Bruce Willis}	{rating=R}	0.010	0.556	1.28	10
5908	{star=Bruce Willis}	{score=average}	0.011	0.611	1.25	11
5909	{star=Bruce Willis}	{gross=average}	0.013	0.722	1.45	13
5910	{star=Bruce Willis}	{budget=average}	0.014	0.778	1.55	14
5911	{star=Bruce Willis}	{directorPopularity=average}	0.012	0.667	1.30	12
5912	{star=Bruce Willis}	{runtime=average}	0.012	0.667	1.26	12
5838	{star=Robert De Niro}	{rating=R}	0.012	0.800	1.85	12
5839	{star=Robert De Niro}	{votes=average}	0.011	0.733	1.47	11
5840	{star=Robert De Niro}	{budget=average}	0.012	0.800	1.59	12
5841	{star=Robert De Niro}	{starPopularity=average}	0.015	1.000	1.97	15
5842	{star=Robert De Niro}	{runtime=average}	0.011	0.733	1.38	11
5800	{star=Denzel Washington}	{runtime=long}	0.011	0.579	2.28	11
5801	{star=Denzel Washington}	{percProfit=average}	0.010	0.526	1.49	10
5802	{star=Denzel Washington}	{rating=R}	0.013	0.684	1.58	13
5803	{star=Denzel Washington}	{score=average}	0.010	0.526	1.07	10
5804	{star=Denzel Washington}	{gross=average}	0.015	0.789	1.58	15
5805	{star=Denzel Washington}	{budget=average}	0.011	0.579	1.15	11
5806	{star=Denzel Washington}	{starPopularity=average}	0.019	1.000	1.97	19
5735	{star=Will Smith}	{budget=high}	0.010	0.769	3.06	10
5736	{star=Will Smith}	{score=average}	0.010	0.769	1.57	10
5737	{star=Will Smith}	{starPopularity=average}	0.013	1.000	1.97	13
5670	{star=Arnold Schwarzenegger}	{starPopularity=high}	0.013	1.000	3.99	13
5671	{star=Arnold Schwarzenegger}	{votes=average}	0.010	0.769	1.54	10
5672	{star=Arnold Schwarzenegger}	{directorPopularity=average}	0.010	0.769	1.50	10
5607	{star=Leonardo DiCaprio}	{starPopularity=high}	0.010	1.000	3.99	10

Figure 12.

STARS ON THE LHS, BY STAR

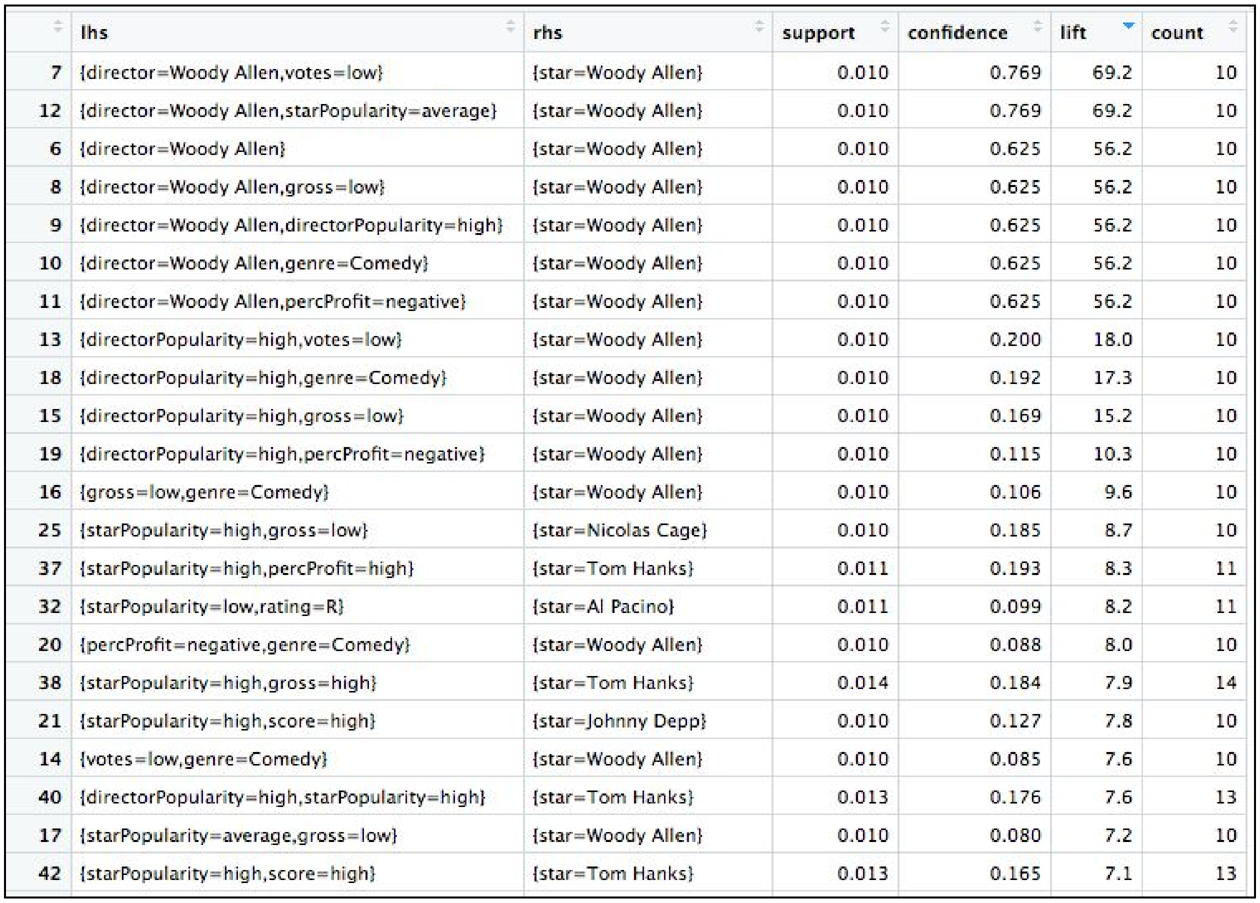
- **TOP 10 RULES FOR LIFT (RHS = All variables)**



	lhs	rhs	support	confidence	lift	count
35	{directorPopularity=average,genre=Animation}	{rating=PG}	0.011	0.92	7.3	11
34	{runtime=short,genre=Animation}	{rating=PG}	0.012	0.80	6.4	12
159	{gross=low,percProfit=high}	{budget=low}	0.016	1.00	4.1	16
160	{gross=low,percProfit=average}	{budget=low}	0.026	1.00	4.1	26
62	{director=Woody Allen,star=Woody Allen}	{votes=low}	0.010	1.00	4.0	10
63	{star=Woody Allen,gross=low}	{votes=low}	0.010	1.00	4.0	10
64	{directorPopularity=high,star=Woody Allen}	{votes=low}	0.010	1.00	4.0	10
65	{star=Woody Allen,genre=Comedy}	{votes=low}	0.010	1.00	4.0	10
102	{director=Woody Allen}	{gross=low}	0.016	1.00	4.0	16
103	{director=Woody Allen,star=Woody Allen}	{gross=low}	0.010	1.00	4.0	10
104	{star=Woody Allen,votes=low}	{gross=low}	0.010	1.00	4.0	10
105	{directorPopularity=high,star=Woody Allen}	{gross=low}	0.010	1.00	4.0	10
106	{star=Woody Allen,genre=Comedy}	{gross=low}	0.010	1.00	4.0	10
109	{director=Woody Allen,budget=low}	{gross=low}	0.013	1.00	4.0	13
110	{director=Woody Allen,votes=low}	{gross=low}	0.013	1.00	4.0	13
111	{director=Woody Allen,directorPopularity=high}	{gross=low}	0.016	1.00	4.0	16
112	{director=Woody Allen,genre=Comedy}	{gross=low}	0.016	1.00	4.0	16
113	{director=Woody Allen,percProfit=negative}	{gross=low}	0.016	1.00	4.0	16

Figure 13.

- **TOP 10 RULES FOR LIFT (RHS = Stars)**

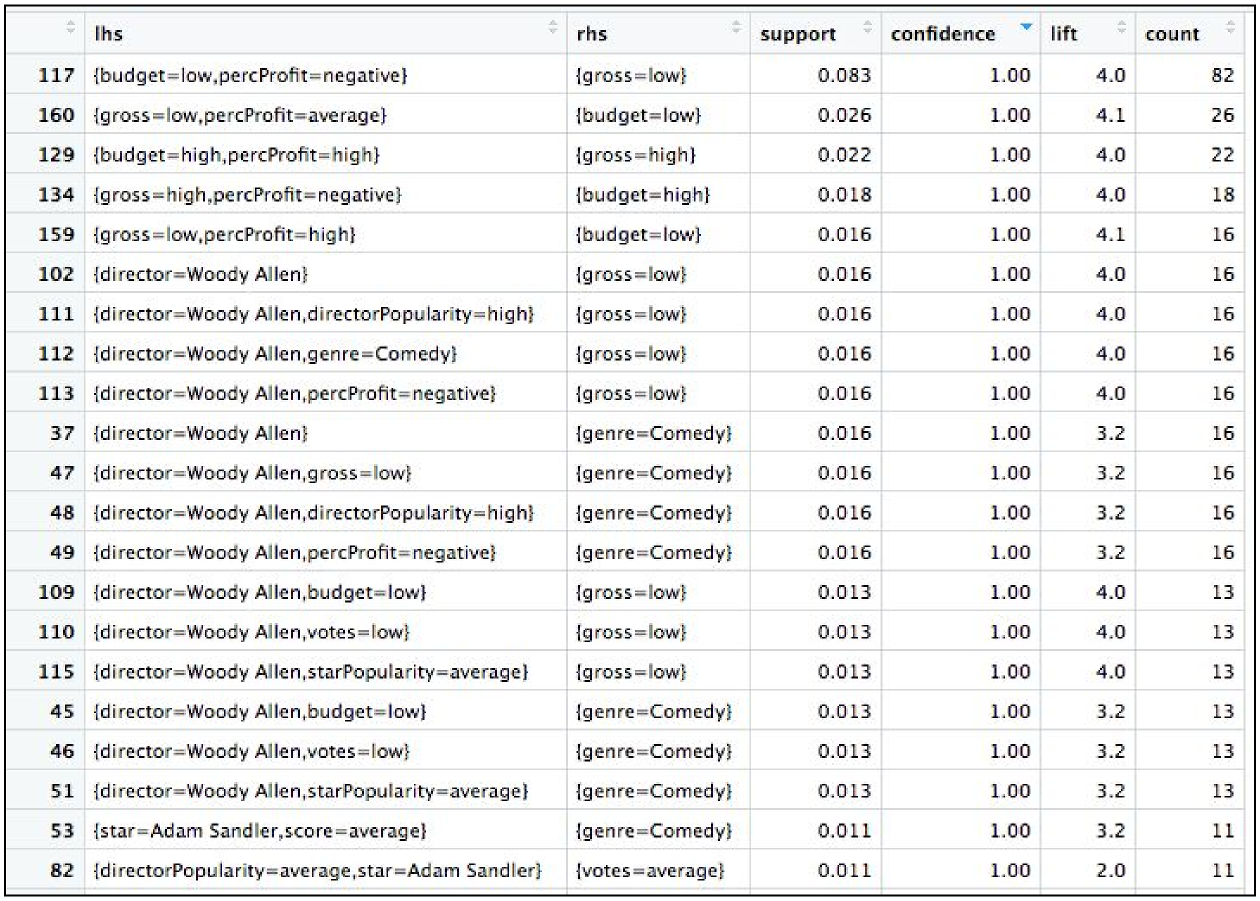


	lhs	rhs	support	confidence	lift	count
7	{director=Woody Allen,votes=low}	{star=Woody Allen}	0.010	0.769	69.2	10
12	{director=Woody Allen,starPopularity=average}	{star=Woody Allen}	0.010	0.769	69.2	10
6	{director=Woody Allen}	{star=Woody Allen}	0.010	0.625	56.2	10
8	{director=Woody Allen,gross=low}	{star=Woody Allen}	0.010	0.625	56.2	10
9	{director=Woody Allen,directorPopularity=high}	{star=Woody Allen}	0.010	0.625	56.2	10
10	{director=Woody Allen,genre=Comedy}	{star=Woody Allen}	0.010	0.625	56.2	10
11	{director=Woody Allen,percProfit=negative}	{star=Woody Allen}	0.010	0.625	56.2	10
13	{directorPopularity=high,votes=low}	{star=Woody Allen}	0.010	0.200	18.0	10
18	{directorPopularity=high,genre=Comedy}	{star=Woody Allen}	0.010	0.192	17.3	10
15	{directorPopularity=high,gross=low}	{star=Woody Allen}	0.010	0.169	15.2	10
19	{directorPopularity=high,percProfit=negative}	{star=Woody Allen}	0.010	0.115	10.3	10
16	{gross=low,genre=Comedy}	{star=Woody Allen}	0.010	0.106	9.6	10
25	{starPopularity=high,gross=low}	{star=Nicolas Cage}	0.010	0.185	8.7	10
37	{starPopularity=high,percProfit=high}	{star=Tom Hanks}	0.011	0.193	8.3	11
32	{starPopularity=low,rating=R}	{star=Al Pacino}	0.011	0.099	8.2	11
20	{percProfit=negative,genre=Comedy}	{star=Woody Allen}	0.010	0.088	8.0	10
38	{starPopularity=high,gross=high}	{star=Tom Hanks}	0.014	0.184	7.9	14
21	{starPopularity=high,score=high}	{star=Johnny Depp}	0.010	0.127	7.8	10
14	{votes=low,genre=Comedy}	{star=Woody Allen}	0.010	0.085	7.6	10
40	{directorPopularity=high,starPopularity=high}	{star=Tom Hanks}	0.013	0.176	7.6	13
17	{starPopularity=average,gross=low}	{star=Woody Allen}	0.010	0.080	7.2	10
42	{starPopularity=high,score=high}	{star=Tom Hanks}	0.013	0.165	7.1	13

Figure 14.

- **TOP 10 RULES FOR CONFIDENCE (RHS = All variables)**

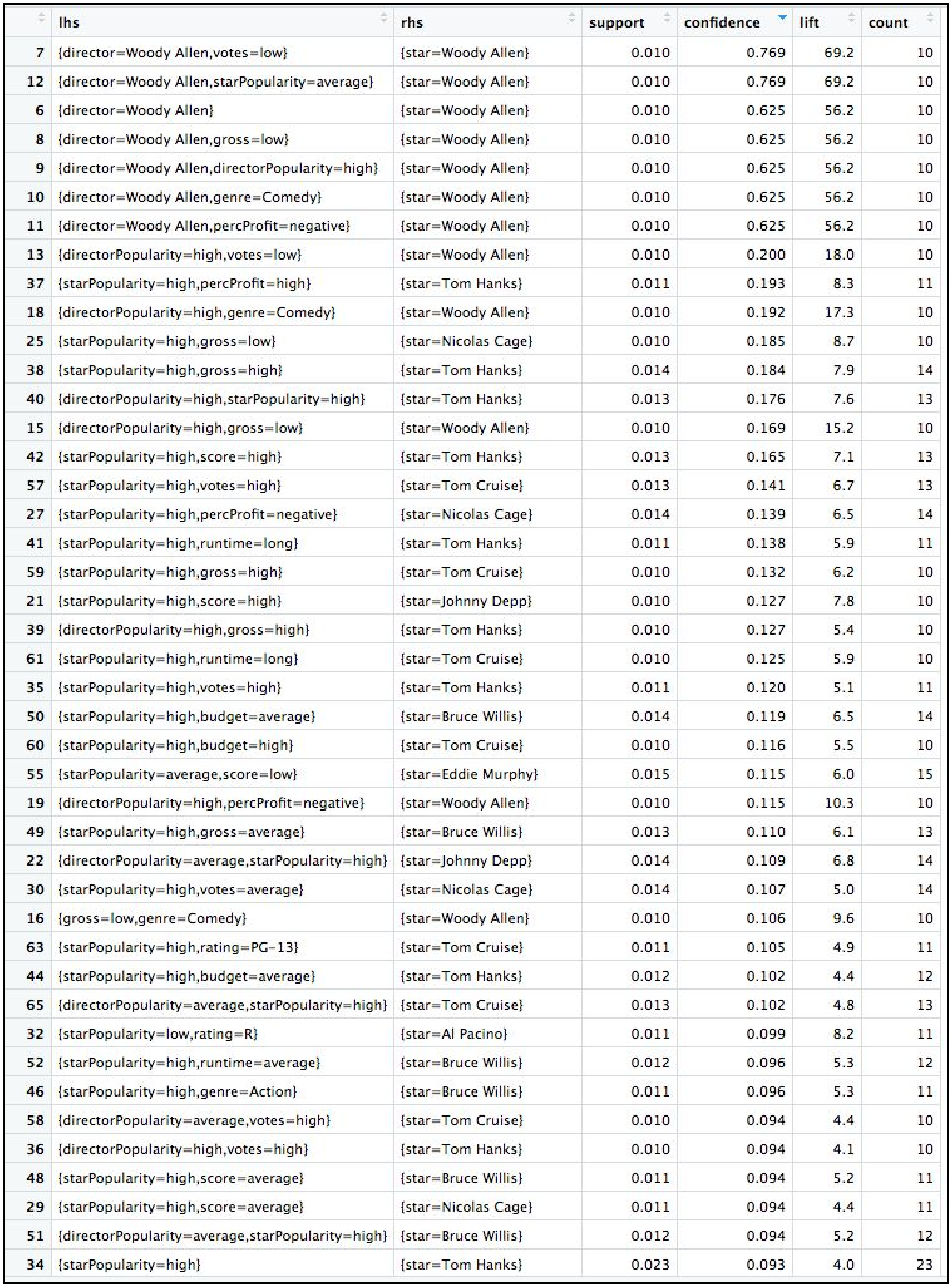
II. *These are unfortunately boring rules as there is a clear (and mathematical) relationship between budget, gross and percent profit. However, we left them in to see how they behave not with one another but how they behave with the other variables.*



	lhs	rhs	support	confidence	lift	count
117	{budget=low,percProfit=negative}	{gross=low}	0.083	1.00	4.0	82
160	{gross=low,percProfit=average}	{budget=low}	0.026	1.00	4.1	26
129	{budget=high,percProfit=high}	{gross=high}	0.022	1.00	4.0	22
134	{gross=high,percProfit=negative}	{budget=high}	0.018	1.00	4.0	18
159	{gross=low,percProfit=high}	{budget=low}	0.016	1.00	4.1	16
102	{director=Woody Allen}	{gross=low}	0.016	1.00	4.0	16
111	{director=Woody Allen,directorPopularity=high}	{gross=low}	0.016	1.00	4.0	16
112	{director=Woody Allen,genre=Comedy}	{gross=low}	0.016	1.00	4.0	16
113	{director=Woody Allen,percProfit=negative}	{gross=low}	0.016	1.00	4.0	16
37	{director=Woody Allen}	{genre=Comedy}	0.016	1.00	3.2	16
47	{director=Woody Allen,gross=low}	{genre=Comedy}	0.016	1.00	3.2	16
48	{director=Woody Allen,directorPopularity=high}	{genre=Comedy}	0.016	1.00	3.2	16
49	{director=Woody Allen,percProfit=negative}	{genre=Comedy}	0.016	1.00	3.2	16
109	{director=Woody Allen,budget=low}	{gross=low}	0.013	1.00	4.0	13
110	{director=Woody Allen,votes=low}	{gross=low}	0.013	1.00	4.0	13
115	{director=Woody Allen,starPopularity=average}	{gross=low}	0.013	1.00	4.0	13
45	{director=Woody Allen,budget=low}	{genre=Comedy}	0.013	1.00	3.2	13
46	{director=Woody Allen,votes=low}	{genre=Comedy}	0.013	1.00	3.2	13
51	{director=Woody Allen,starPopularity=average}	{genre=Comedy}	0.013	1.00	3.2	13
53	{star=Adam Sandler,score=average}	{genre=Comedy}	0.011	1.00	3.2	11
82	{directorPopularity=average,star=Adam Sandler}	{votes=average}	0.011	1.00	2.0	11

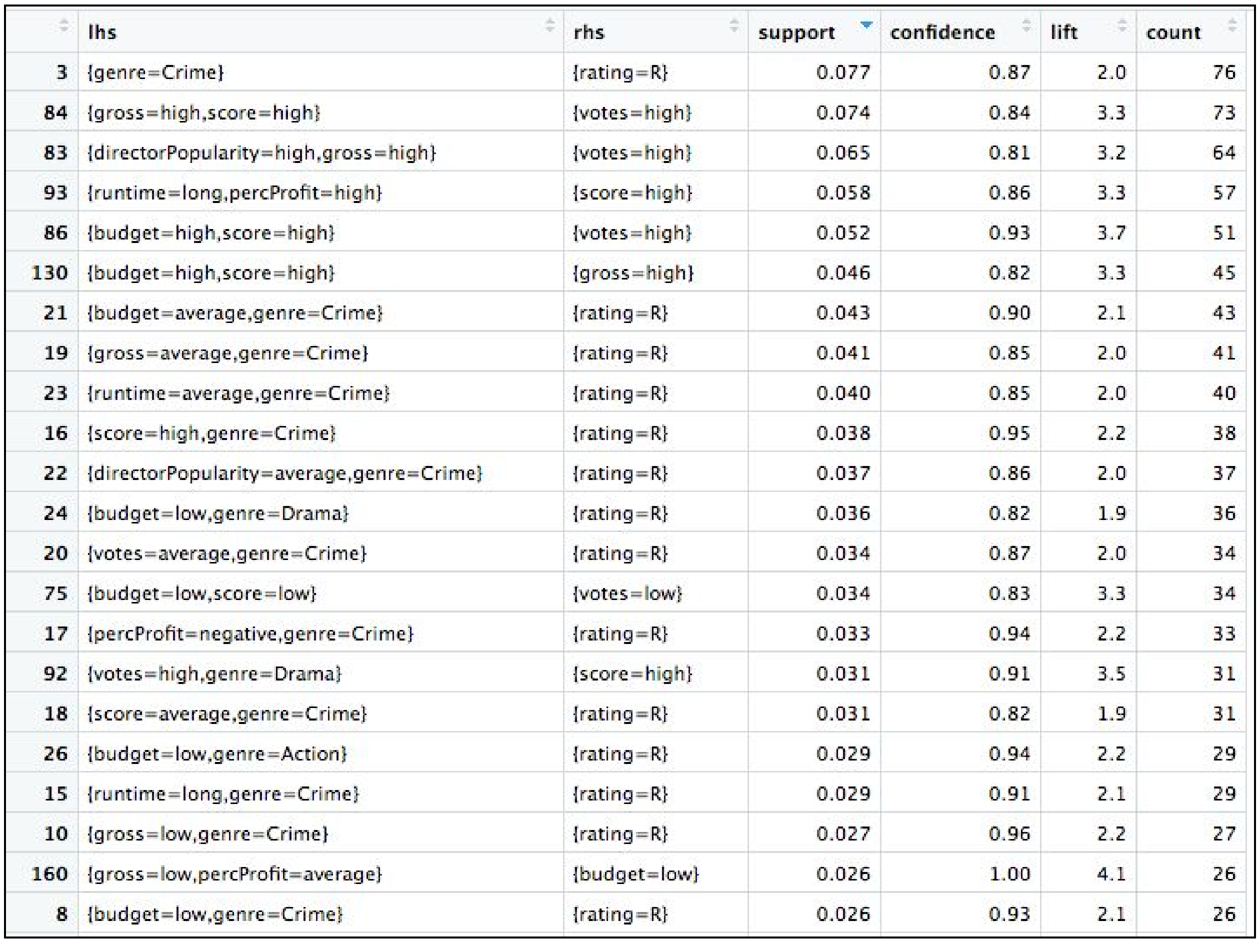
Figure 15.

- **TOP 10 RULES FOR CONFIDENCE (RHS = Stars)**



	lhs	rhs	support	confidence	lift	count
7	{director=Woody Allen,votes=low}	{star=Woody Allen}	0.010	0.769	69.2	10
12	{director=Woody Allen,starPopularity=average}	{star=Woody Allen}	0.010	0.769	69.2	10
6	{director=Woody Allen}	{star=Woody Allen}	0.010	0.625	56.2	10
8	{director=Woody Allen,gross=low}	{star=Woody Allen}	0.010	0.625	56.2	10
9	{director=Woody Allen,directorPopularity=high}	{star=Woody Allen}	0.010	0.625	56.2	10
10	{director=Woody Allen,genre=Comedy}	{star=Woody Allen}	0.010	0.625	56.2	10
11	{director=Woody Allen,percProfit=negative}	{star=Woody Allen}	0.010	0.625	56.2	10
13	{directorPopularity=high,votes=low}	{star=Woody Allen}	0.010	0.200	18.0	10
37	{starPopularity=high,percProfit=high}	{star=Tom Hanks}	0.011	0.193	8.3	11
18	{directorPopularity=high,genre=Comedy}	{star=Woody Allen}	0.010	0.192	17.3	10
25	{starPopularity=high,gross=low}	{star=Nicolas Cage}	0.010	0.185	8.7	10
38	{starPopularity=high,gross=high}	{star=Tom Hanks}	0.014	0.184	7.9	14
40	{directorPopularity=high,starPopularity=high}	{star=Tom Hanks}	0.013	0.176	7.6	13
15	{directorPopularity=high,gross=low}	{star=Woody Allen}	0.010	0.169	15.2	10
42	{starPopularity=high,score=high}	{star=Tom Hanks}	0.013	0.165	7.1	13
57	{starPopularity=high,votes=high}	{star=Tom Cruise}	0.013	0.141	6.7	13
27	{starPopularity=high,percProfit=negative}	{star=Nicolas Cage}	0.014	0.139	6.5	14
41	{starPopularity=high,runtime=long}	{star=Tom Hanks}	0.011	0.138	5.9	11
59	{starPopularity=high,gross=high}	{star=Tom Cruise}	0.010	0.132	6.2	10
21	{starPopularity=high,score=high}	{star=Johnny Depp}	0.010	0.127	7.8	10
39	{directorPopularity=high,gross=high}	{star=Tom Hanks}	0.010	0.127	5.4	10
61	{starPopularity=high,runtime=long}	{star=Tom Cruise}	0.010	0.125	5.9	10
35	{starPopularity=high,votes=high}	{star=Tom Hanks}	0.011	0.120	5.1	11
50	{starPopularity=high,budget=average}	{star=Bruce Willis}	0.014	0.119	6.5	14
60	{starPopularity=high,budget=high}	{star=Tom Cruise}	0.010	0.116	5.5	10
55	{starPopularity=average,score=low}	{star=Eddie Murphy}	0.015	0.115	6.0	15
19	{directorPopularity=high,percProfit=negative}	{star=Woody Allen}	0.010	0.115	10.3	10
49	{starPopularity=high,gross=average}	{star=Bruce Willis}	0.013	0.110	6.1	13
22	{directorPopularity=average,starPopularity=high}	{star=Johnny Depp}	0.014	0.109	6.8	14
30	{starPopularity=high,votes=average}	{star=Nicolas Cage}	0.014	0.107	5.0	14
16	{gross=low,genre=Comedy}	{star=Woody Allen}	0.010	0.106	9.6	10
63	{starPopularity=high,rating=PG-13}	{star=Tom Cruise}	0.011	0.105	4.9	11
44	{starPopularity=high,budget=average}	{star=Tom Hanks}	0.012	0.102	4.4	12
65	{directorPopularity=average,starPopularity=high}	{star=Tom Cruise}	0.013	0.102	4.8	13
32	{starPopularity=low,rating=R}	{star=Al Pacino}	0.011	0.099	8.2	11
52	{starPopularity=high,runtime=average}	{star=Bruce Willis}	0.012	0.096	5.3	12
46	{starPopularity=high,genre=Action}	{star=Bruce Willis}	0.011	0.096	5.3	11
58	{directorPopularity=average,votes=high}	{star=Tom Cruise}	0.010	0.094	4.4	10
36	{directorPopularity=high,votes=high}	{star=Tom Hanks}	0.010	0.094	4.1	10
48	{starPopularity=high,score=average}	{star=Bruce Willis}	0.011	0.094	5.2	11
29	{starPopularity=high,score=average}	{star=Nicolas Cage}	0.011	0.094	4.4	11
51	{directorPopularity=average,starPopularity=high}	{star=Bruce Willis}	0.012	0.094	5.2	12
34	{starPopularity=high}	{star=Tom Hanks}	0.023	0.093	4.0	23

- **TOP 10 RULES FOR SUPPORT (RHS = All variables)**



	lhs	rhs	support	confidence	lift	count
3	{genre=Crime}	{rating=R}	0.077	0.87	2.0	76
84	{gross=high,score=high}	{votes=high}	0.074	0.84	3.3	73
83	{directorPopularity=high,gross=high}	{votes=high}	0.065	0.81	3.2	64
93	{runtime=long,percProfit=high}	{score=high}	0.058	0.86	3.3	57
86	{budget=high,score=high}	{votes=high}	0.052	0.93	3.7	51
130	{budget=high,score=high}	{gross=high}	0.046	0.82	3.3	45
21	{budget=average,genre=Crime}	{rating=R}	0.043	0.90	2.1	43
19	{gross=average,genre=Crime}	{rating=R}	0.041	0.85	2.0	41
23	{runtime=average,genre=Crime}	{rating=R}	0.040	0.85	2.0	40
16	{score=high,genre=Crime}	{rating=R}	0.038	0.95	2.2	38
22	{directorPopularity=average,genre=Crime}	{rating=R}	0.037	0.86	2.0	37
24	{budget=low,genre=Drama}	{rating=R}	0.036	0.82	1.9	36
20	{votes=average,genre=Crime}	{rating=R}	0.034	0.87	2.0	34
75	{budget=low,score=low}	{votes=low}	0.034	0.83	3.3	34
17	{percProfit=negative,genre=Crime}	{rating=R}	0.033	0.94	2.2	33
92	{votes=high,genre=Drama}	{score=high}	0.031	0.91	3.5	31
18	{score=average,genre=Crime}	{rating=R}	0.031	0.82	1.9	31
26	{budget=low,genre=Action}	{rating=R}	0.029	0.94	2.2	29
15	{runtime=long,genre=Crime}	{rating=R}	0.029	0.91	2.1	29
10	{gross=low,genre=Crime}	{rating=R}	0.027	0.96	2.2	27
160	{gross=low,percProfit=average}	{budget=low}	0.026	1.00	4.1	26
8	{budget=low,genre=Crime}	{rating=R}	0.026	0.93	2.1	26

Figure 16.

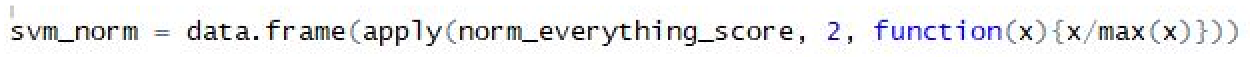
III. K-MEANS CLUSTERING & HCLUST

SCORE

kmeans()

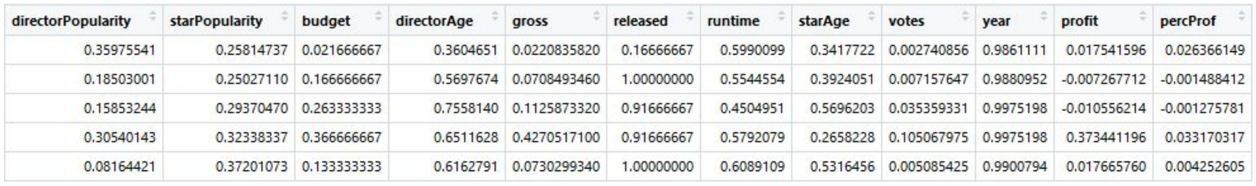
Model 1

The first model utilized a normalized dataset of all numeric variables with the score variable removed. The data frame contains 741 observations of 12 variables. The data was normalized by dividing every value each column by the maximum value in the column. This resulted with values between 0 and 1.



```
svm_norm = data.frame(apply(norm_everything_score, 2, function(x){x/max(x)}))
```

Figure 16a. Code Snippet to Normalize Dataset

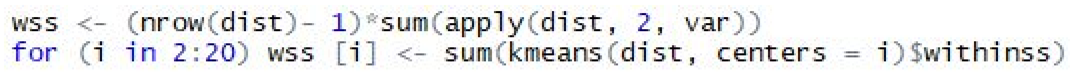


directorPopularity	starPopularity	budget	directorAge	gross	released	runtime	starAge	votes	year	profit	percProf
0.35975541	0.25814737	0.021666667	0.3604651	0.0220835820	0.16666667	0.5990099	0.3417722	0.002740856	0.9861111	0.017541596	0.026366149
0.18503001	0.25027110	0.166666667	0.5697674	0.0708493460	1.00000000	0.5544554	0.3924051	0.007157647	0.9880952	-0.007267712	-0.001488412
0.15853244	0.29370470	0.263333333	0.7558140	0.1125873320	0.91666667	0.4504951	0.5696203	0.035359331	0.9975198	-0.010556214	-0.001275781
0.30540143	0.32338337	0.366666667	0.6511628	0.4270517100	0.91666667	0.5792079	0.2658228	0.105067975	0.9975198	0.373441196	0.033170317
0.08164421	0.37201073	0.133333333	0.6162791	0.0730299340	1.00000000	0.6089109	0.5316456	0.005085425	0.9900794	0.017665760	0.004252605

Figure 17. First 5 Rows of the norm_everything_kMeans df

The data frame was then converted into a cosine distance matrix for the first model.

The elbow method was used to determine the best range of k values.



```
wss <- (nrow(dist)- 1)*sum(apply(dist, 2, var))
for (i in 2:20) wss [i] <- sum(kmeans(dist, centers = i)$withinss)
```

Figure 18. Code Snippet to Determine the best k value

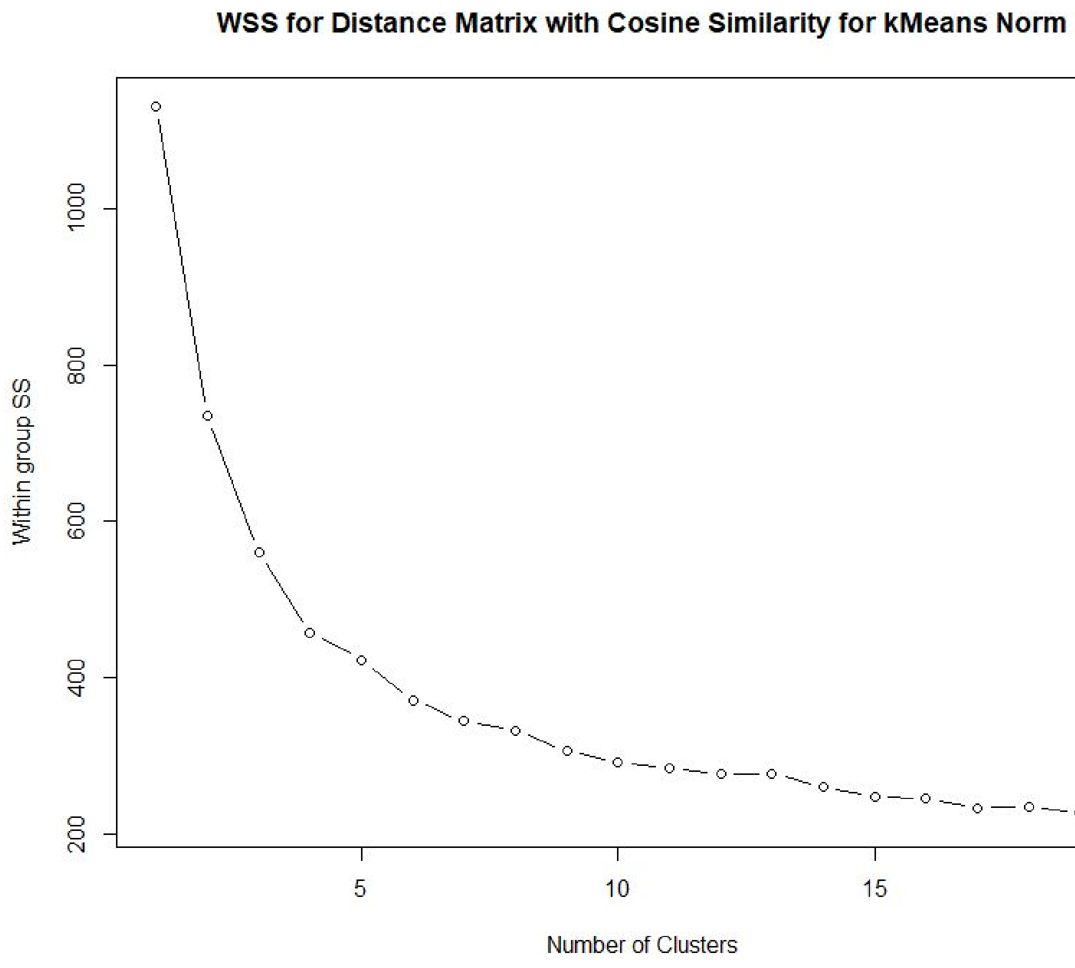


Figure 19. Elbow Method for Distance Matrix DF

The best range of clusters appears to be between 4 - 6 . The first attempt with kmeans had a k of 4 and a nstart of 50.

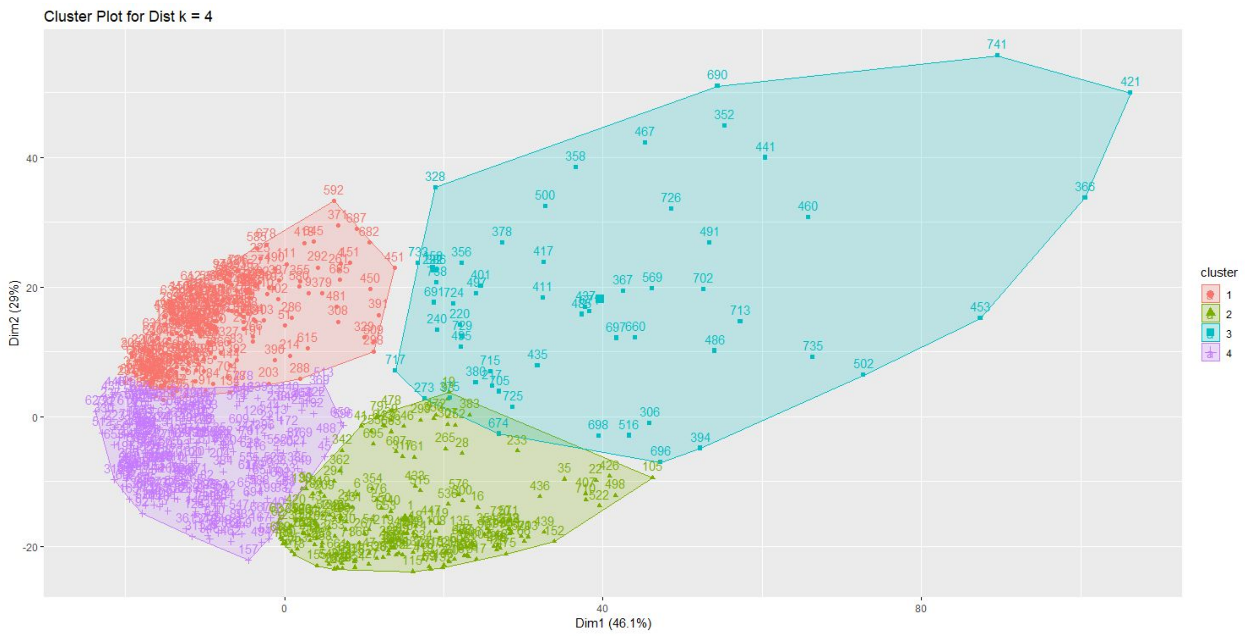
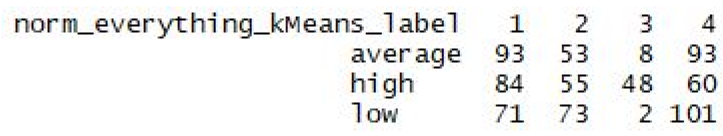


Figure 20. 4 Cluster Plot of kmeans with Cosine Distance Matrix

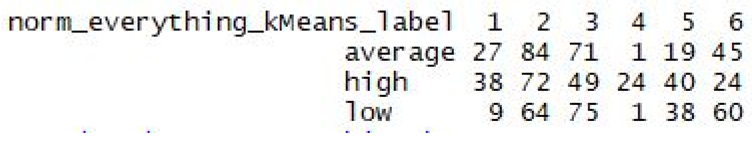


```
 norm_everything_kMeans_label   1   2   3   4
                      average  93  53   8  93
                      high     84  55  48  60
                      low      71  73   2 101
```

Figure 21. Table of Results for kmeans k = 4 Cosine Distance Matrix

The first attempt did not appear to appropriately cluster the movies by score. Cluster 3 had the most success by clustering 48 high scores together with only 8 average scores and 2 low scores. The other 3 clusters did not have the same element of success. The within cluster sum of squares are 108, 121, 116, and 113.

The second attempt had a k of 6 and a nstart of 50.



```
norm_everything_kMeans_label  1  2  3  4  5  6
                     average 27 84 71  1 19 45
                     high    38 72 49 24 40 24
                     low      9 64 75  1 38 60
```

Figure 22. Table of Results for kmeans k = 6 Cosine DIstance Matrix

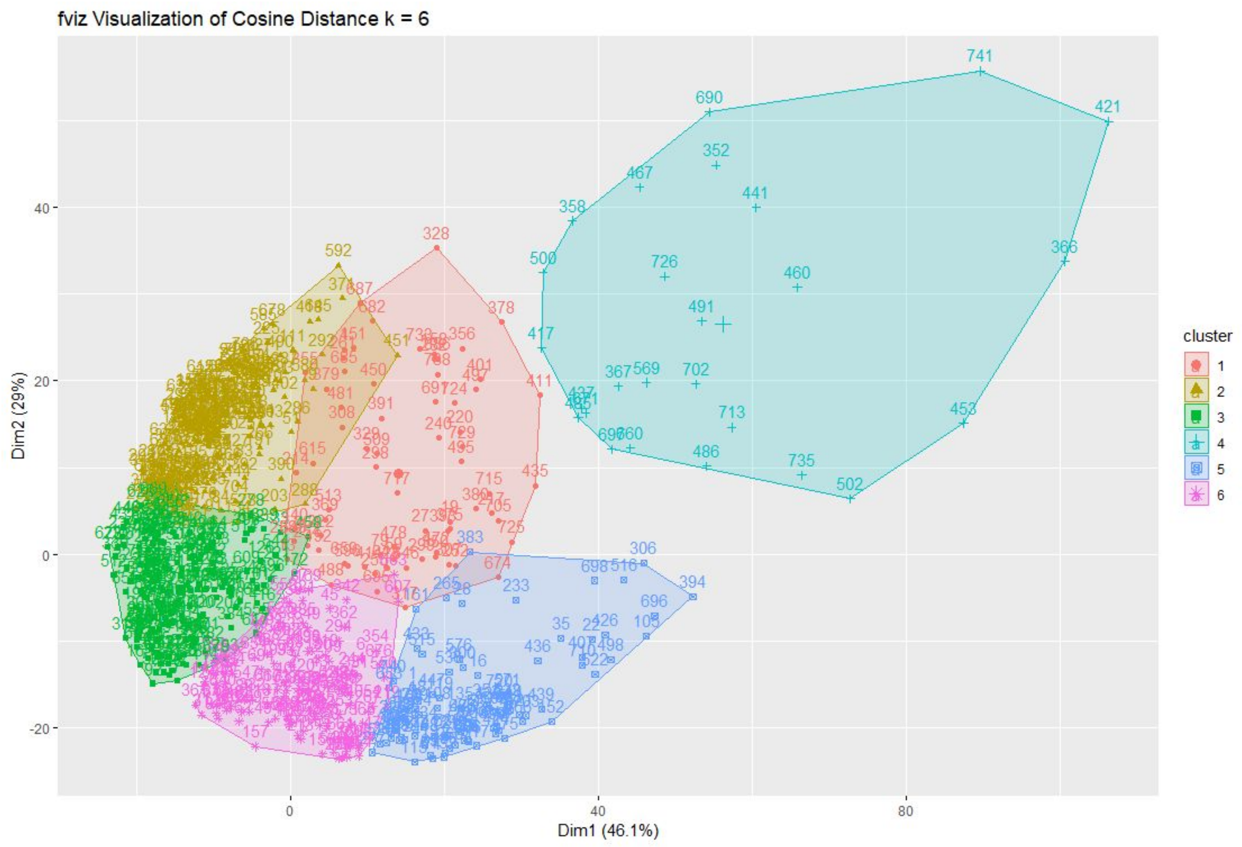
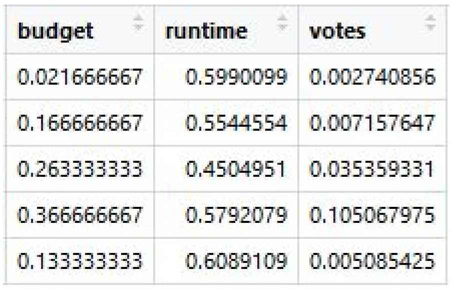


Figure 23. 6 Cluster Plot of kmeans with Cosine Distance Matrix

The second attempt did not net significantly better results. Cluster 4 clustered 24 high scores with only 1 average and 1 low score. Cluster 1 also clustered mainly average and high scores, with only 9 low scores. The within cluster sum of squares by cluster are: 68, 79, 65, 49, 58, and 53. Clustering with all variables is not proving to be beneficial.

Model 2

The second model utilizes the distance matrix from model 1, but the dimensions are reduced to include only votes, budget, runtime and profit. Figure 24 is a visual of the dataframe before the cosine distance matrix transformation.



budget	runtime	votes
0.021666667	0.5990099	0.002740856
0.166666667	0.5544554	0.007157647
0.263333333	0.4504951	0.035359331
0.366666667	0.5792079	0.105067975
0.133333333	0.6089109	0.005085425

Figure 24. First 5 Rows of the reduced_norm_everything_kMeans df

The elbow method was used to determine the best range of k values.

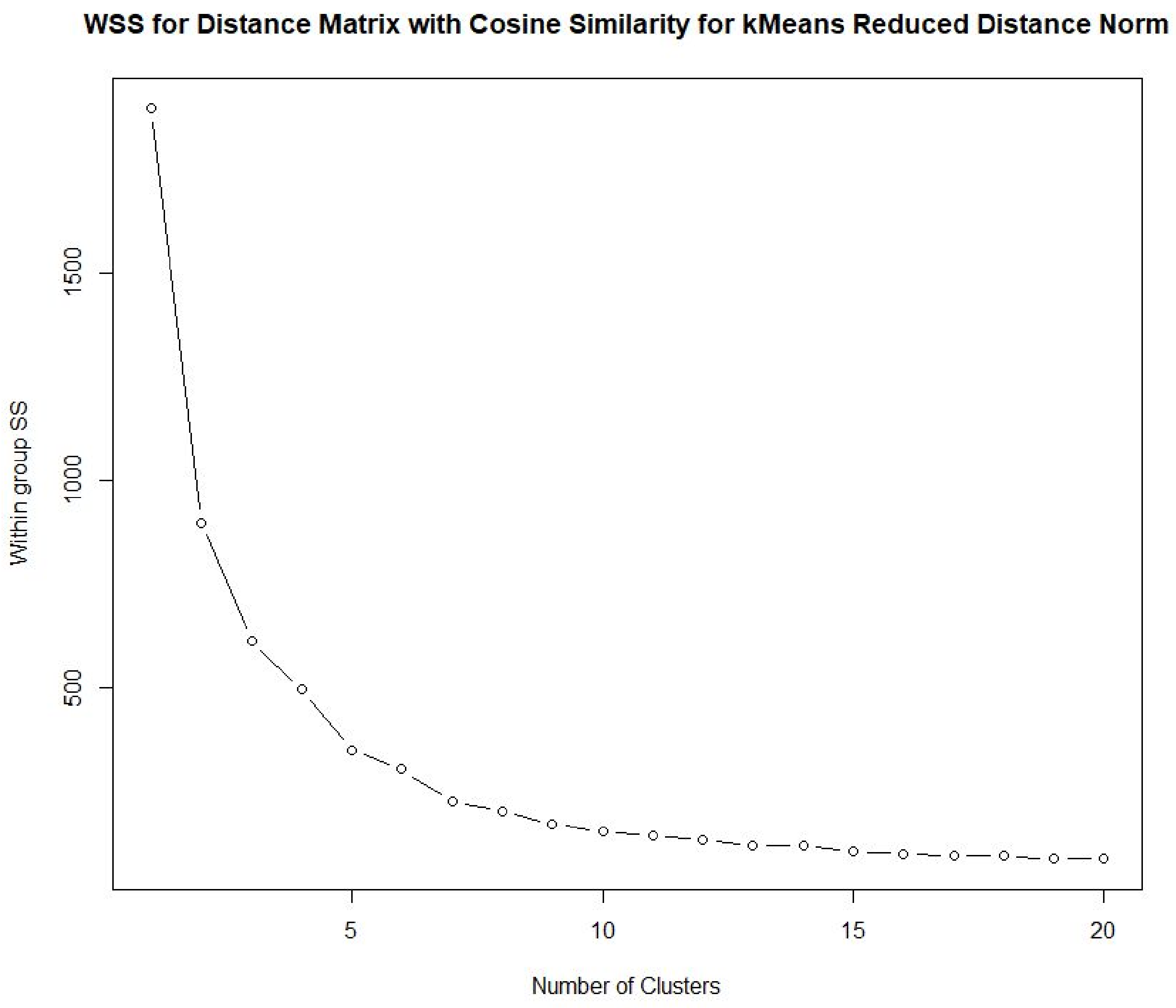


Figure 25. Elbow Method for Reduced Distance Matrix DF

The best range of clusters appears to be between 4 - 6 . The first attempt with kmeans had a k of 4 and a nstart of 50.

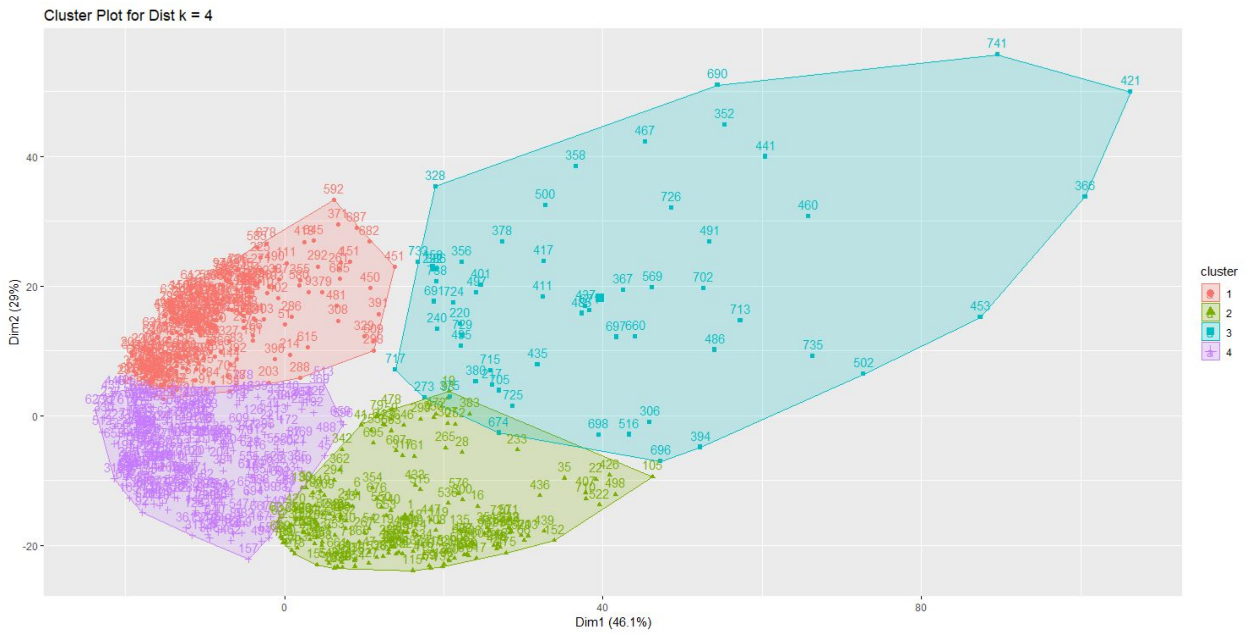
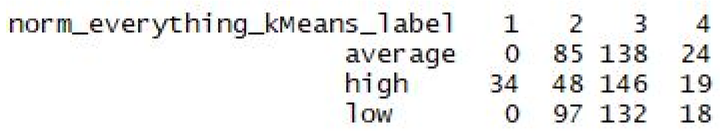


Figure 26. 4 Cluster Plot of kmeans with Reduced Cosine Distance Matrix



```
norm_everything_kMeans_label   1   2   3   4
                     average   0  85 138  24
                     high     34  48 146  19
                     low       0  97 132  18
```

Figure 27. Table of Results for kmeans k = 4 Reduced Cosine DIstance Matrix

The model successfully classified only high scores in cluster 1. Cluster 2 is comprised of 182 movies with a low or average score and only 48 movies with a high score. The 79% of the entries are average or low. However, the other two clusters do not appear to successfully cluster the scores. The within cluster sum of squares by cluster are 122, 103, 146, and 84.

The second attempt had a k of 6 and a nstart of 50.

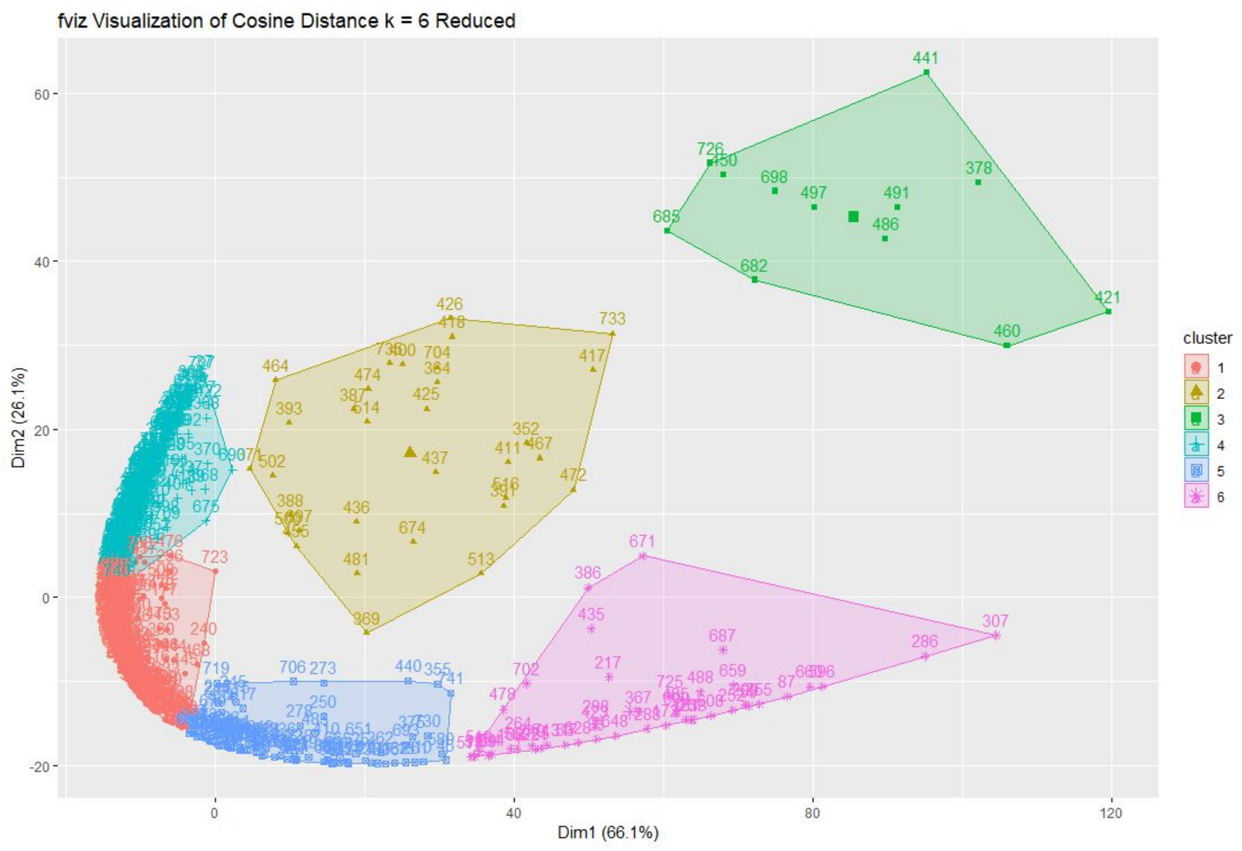
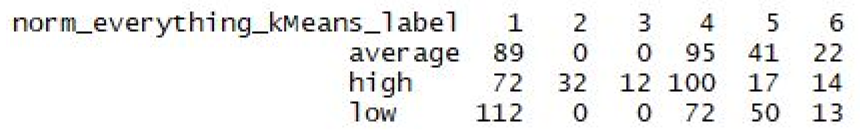


Figure 28. 6 Cluster Plot of kmeans with Reduced Cosine Distance Matrix



```
norm_everything_kMeans_label   1   2   3   4   5   6
                     average  89   0   0  95  41  22
                     high     72  32  12 100  17  14
                     low     112   0   0  72  50  13
```

Figure 29. Table of Results for kmeans k = 6 Reduced Cosine DIstance Matrix

This models appears to have slightly better results than the previous model. Clusters 2 and 3 are comprised entirely of high scores. The within cluster sum of squares by cluster are: 62, 29, 16, 61, 49, and 55.

Model 3

The third model used the reduced dataset from model two without the cosine distance matrix.

The elbow method was used to determine the best range of k values.

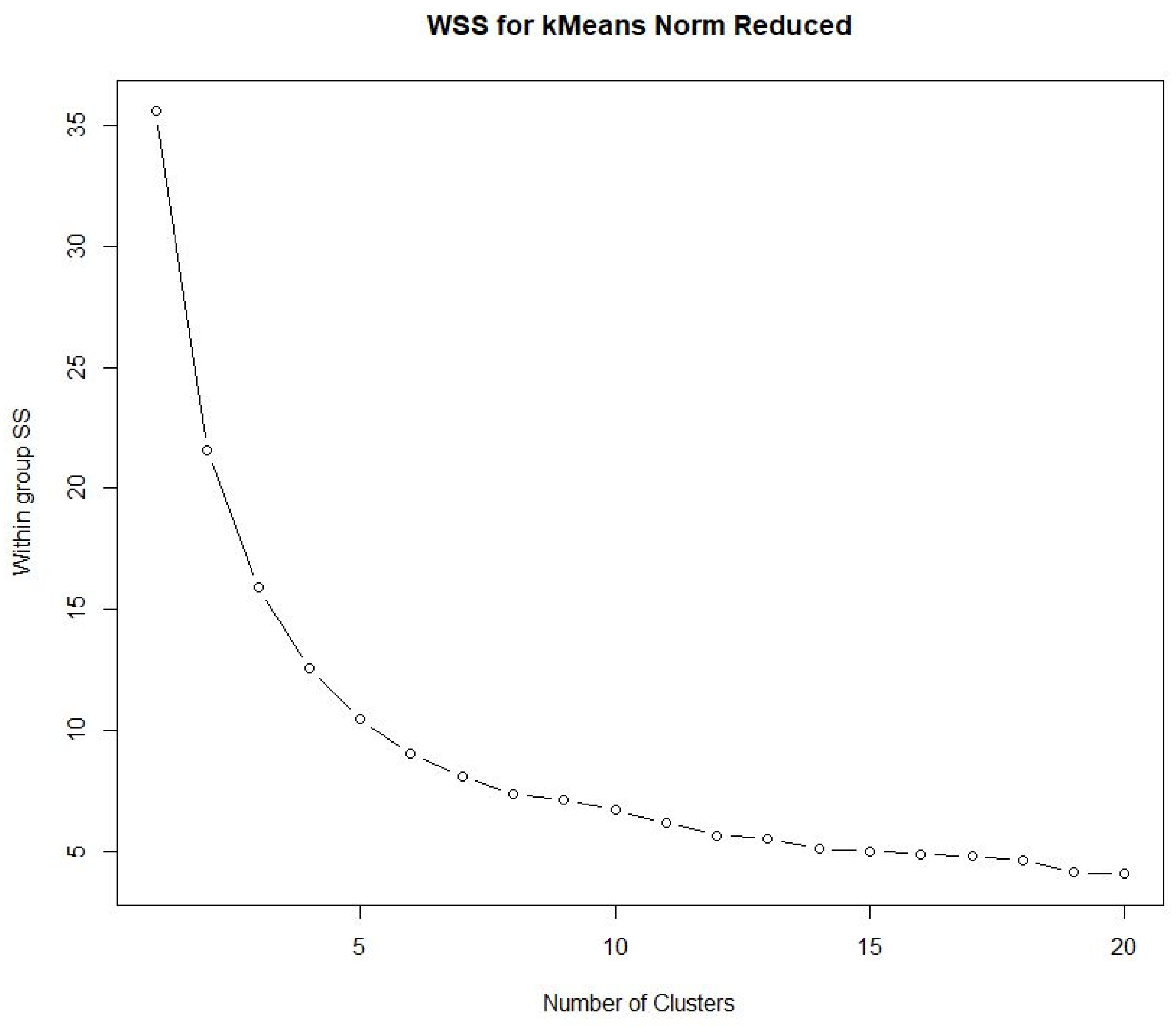


Figure 30. Elbow Method for Reduced Distance Matrix DF

The best range of clusters appears to be between 4 - 6 . The first attempt with kmeans had a k of 4 and a nstart of 50.

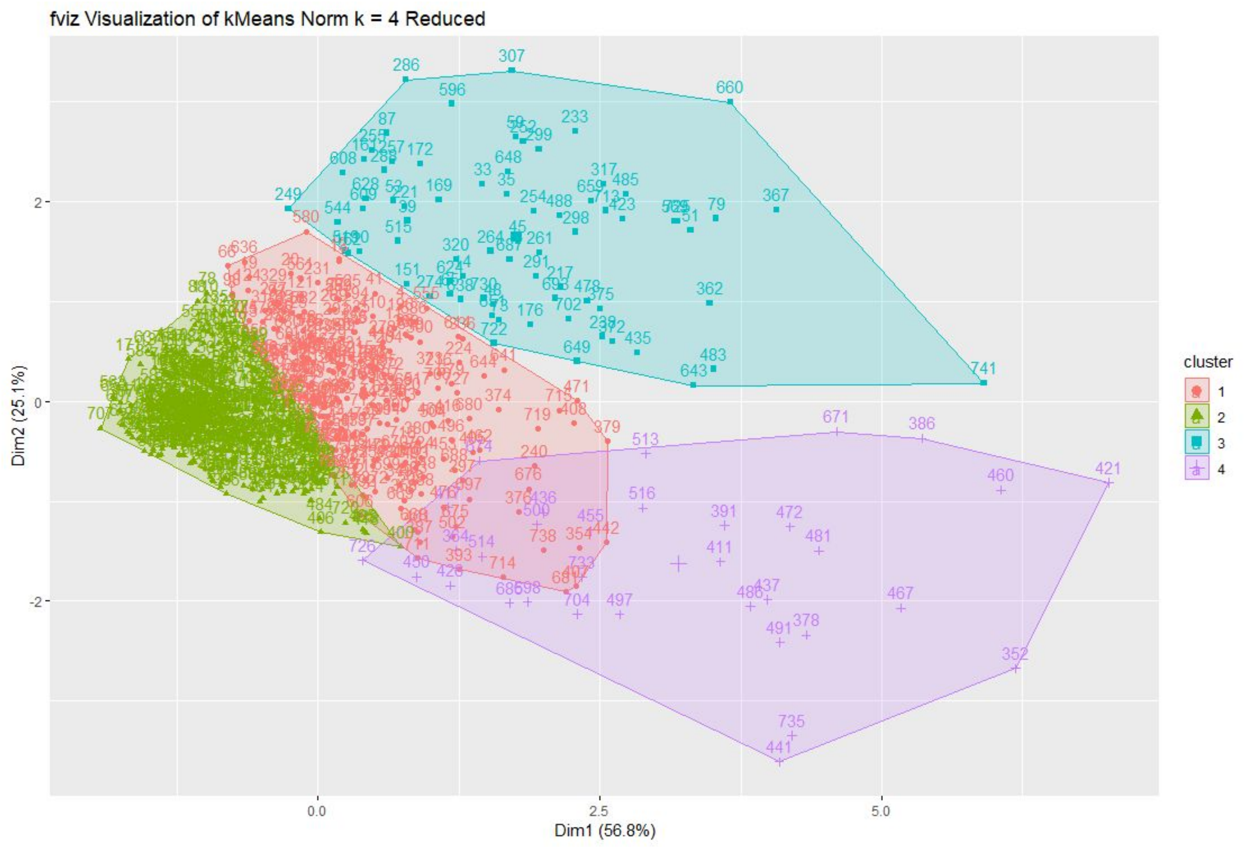


Figure 31. 4 Cluster Plot of kmeans with Reduced everything_Movies USA

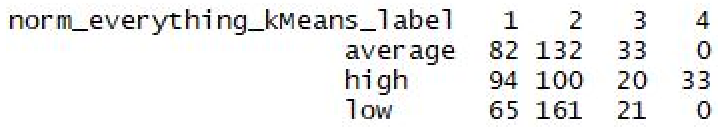

```
norm_everything_kMeans_label   1   2   3   4
                     average  82 132  33   0
                     high     94 100  20  33
                     low      65 161  21   0
```

Figure 32. Table of Results for kmeans k = 4 Reduced everything_MoviesUSA

Cluster 4 is comprised entirely of high scores. The other clusters are a combination of low, average and high scores. The within cluster sum of squares by cluster are: 5, 4, 2, and 2.

The second attempt had a k of 5 and a nstart of 50.

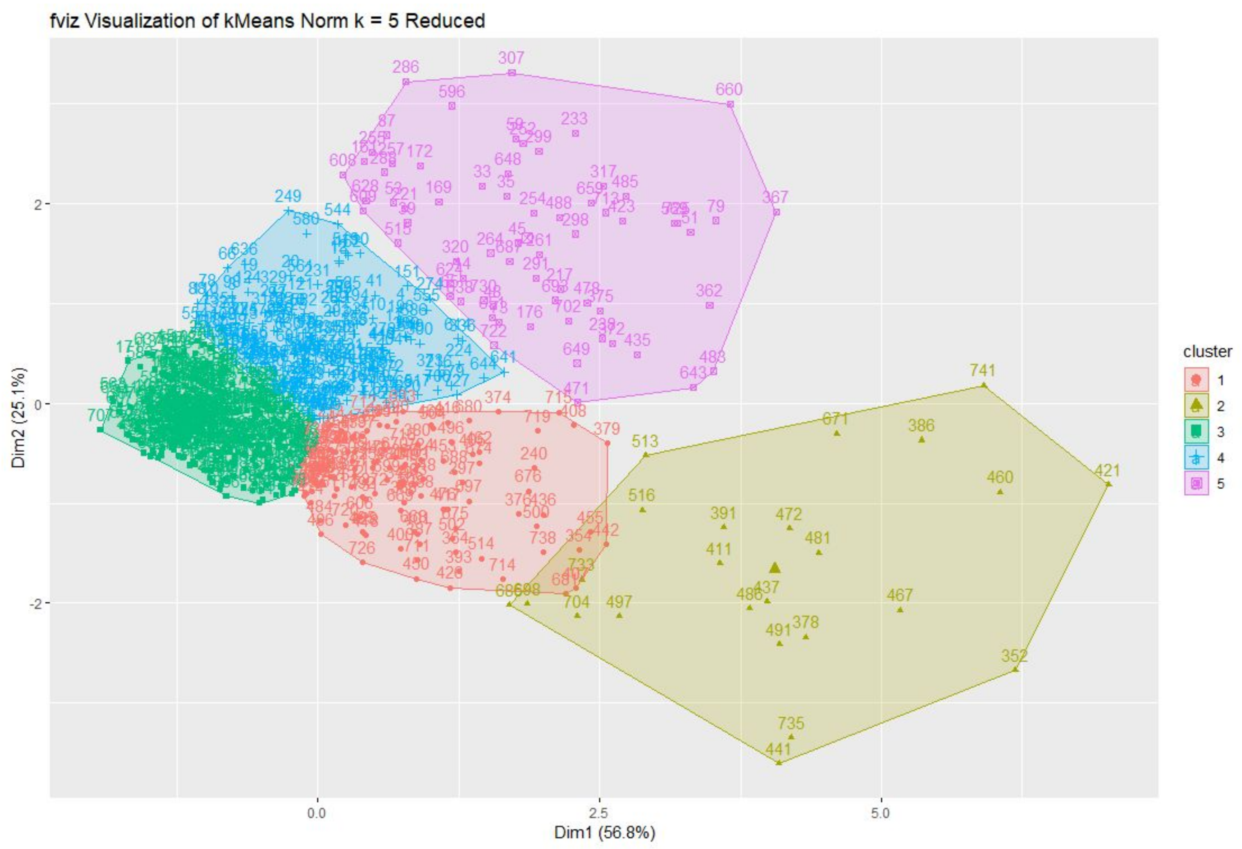


Figure 33. 5 Cluster Plot of kmeans with Reduced everything_Movies USA

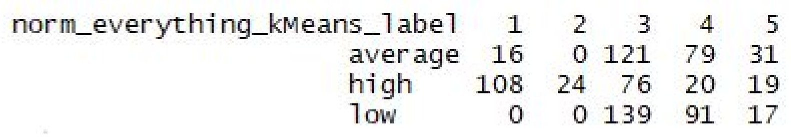

```
norm_everything_kMeans_label   1   2   3   4   5
                     average  16   0 121  79  31
                     high    108  24  76  20  19
                     low       0   0 139  91  17
```

Figure 34. Table of Results for kmeans k = 5 Reduced everything_MoviesUSA

Clusters 1 and 2 successfully classified high scores. Cluster 1 included only 16 average scores with 108 high scores and cluster 2 contained 24 high scores. The other three clusters had more variability between the distribution of scores. However, appears to decently classify average and low scores together. The within cluster sum of squares by cluster are: 3, 2, 2, 2, 2. Ultimately, clustering has not provided especially fruitful. More data needs to be included to yield better results when clustering for score.

PERCENT PROFIT

Toto, I've got a feeling we're not in Kansas anymore.

Dorothy is correct. This is not Kansas. This is N-dimensional space. Within this N-Dimensional space, the munchkins in the machines attempted to cluster percent profit into meaningful groups... and failed miserably. This model utilized the normalized dataset. (NOTE: Because clustering is an unsupervised learning method, the data was not broken into test and training sets for this model). The profit and gross columns were removed. The label was saved and work commenced. The first task was to lure the best number of clusters from the dataset. This involved using WSS and produced the elbow graph as shown in **Figure 35** below. Then a distance matrix with cosine similarity was created and these graphs were born.

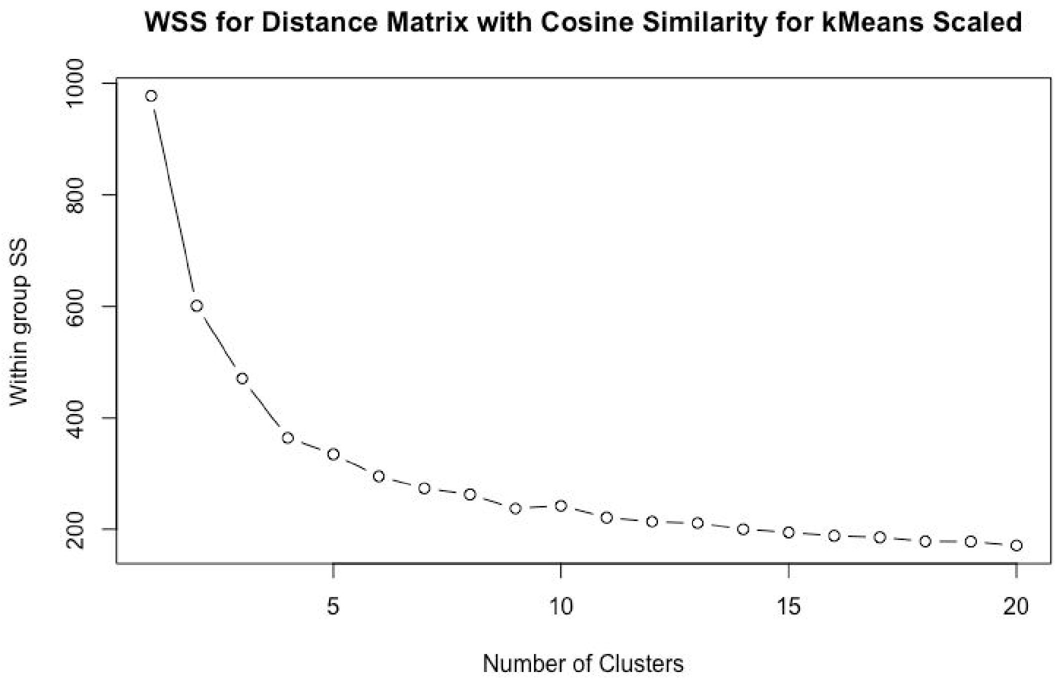


Figure 35.

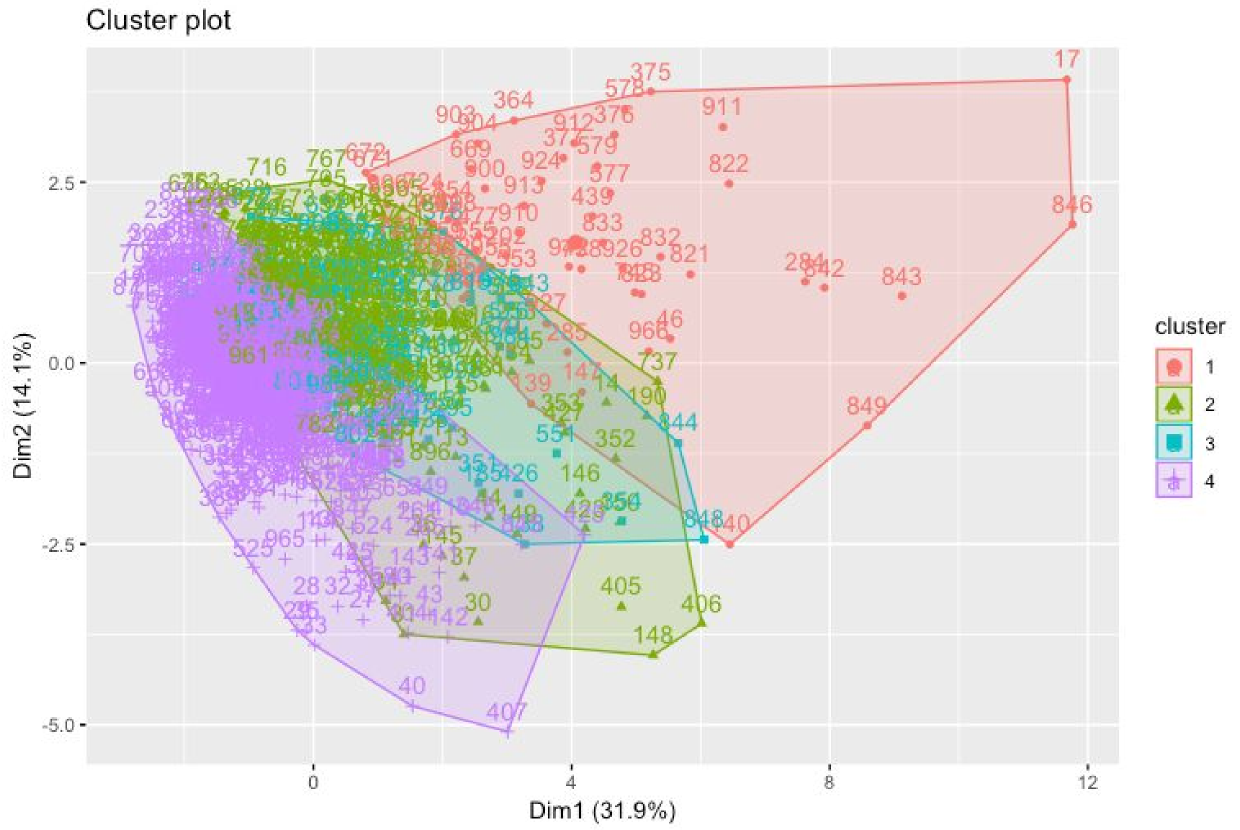


Figure 36.

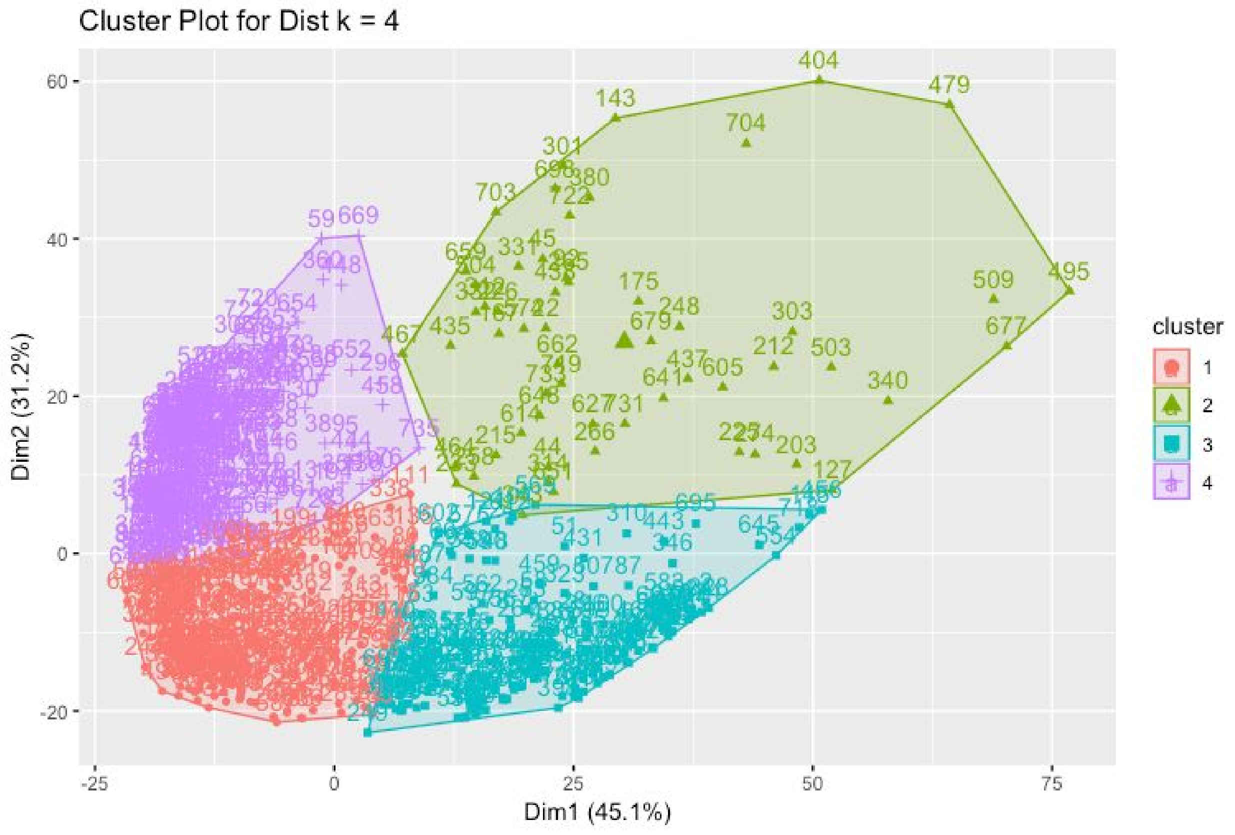


Figure 37

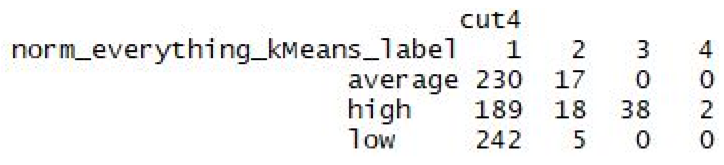
In summation, percent profit can’t be meaningfully clustered with the data on hand.

hclust()

Model

The hclust model used model 2, the reduced cosine distance matrix from kmeans.

The first attempt at hclust used a method of average and a cutree of 4.

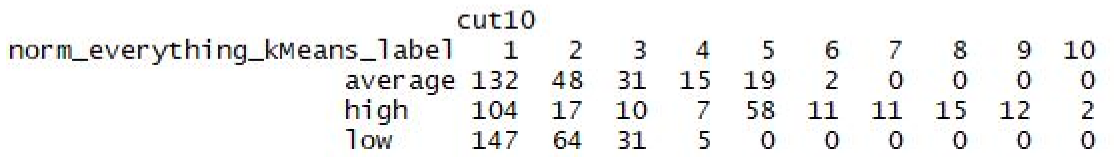


```
                          cut4
norm_everything_kMeans_label   1   2   3   4
                     average 230  17   0   0
                     high    189  18  38   2
                     low     242   5   0   0
```

Figure 38. Table of Results for hclust k = 4 Reduced Distance Cosine Matrix

The model clustered the majority of the scores in cluster one, however clusters 3 and 4 are comprised solely movies with a high score. Cluster 2 is comprised of mainly average and high scores with 5 low scores.

The second hclust model used the reduced cosine distance and a cutree of 10.



```
                           cut10
norm_everything_kMeans_label   1   2   3   4   5   6   7   8   9  10
                     average 132  48  31  15  19   2   0   0   0   0
                     high    104  17  10   7  58  11  11  15  12   2
                     low     147  64  31   5   0   0   0   0   0   0
```

Figure 39. Table of Results for hclust k = 10 Reduced Distance Cosine Matrix

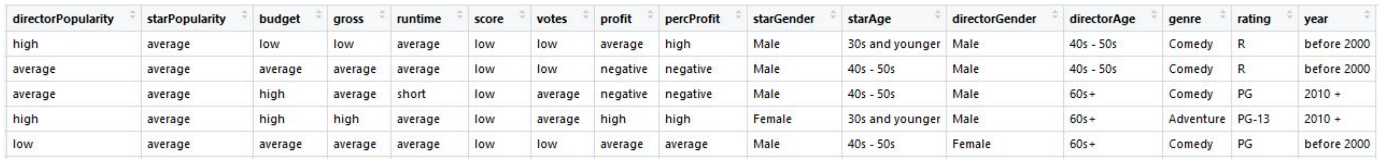
The model clustered the majority of movies in cluster 1. However, clusters 6 - 10 were successful in classifying the high scores and cluster 5 successful scores as average and high.

IV. DECISION TREES

SCORE

Model 1

The first model for the decision tree utilized a fully discretized data frame, with 741 observations of 16 variables. The variables for title, director, star and released were removed from the original discretized dataset.



directorPopularity	starPopularity	budget	gross	runtime	score	votes	profit	percProfit	starGender	starAge	directorGender	directorAge	genre	rating	year
high	average	low	low	average	low	low	average	high	Male	30s and younger	Male	40s - 50s	Comedy	R	before 2000
average	average	average	average	average	low	low	negative	negative	Male	40s - 50s	Male	40s - 50s	Comedy	R	before 2000
average	average	high	average	short	low	average	negative	negative	Male	40s - 50s	Male	60s+	Comedy	PG	2010 +
high	average	high	high	average	low	average	high	high	Female	30s and younger	Male	60s+	Adventure	PG-13	2010 +
low	average	average	average	average	low	low	average	average	Male	40s - 50s	Female	60s+	Comedy	PG	before 2000

Figure 40. First 5 Rows of the dt_discretized_train_score df

The data frame was then split into a training and test dataset. The training dataset contained 519 observations. There were 173 high scores, 173 average scores and 173 low scores. The testing dataset was comprised on 74 high scores, 74 average scores, and 74 low scores for a total of 222 observations. The score label was removed from the testing set.

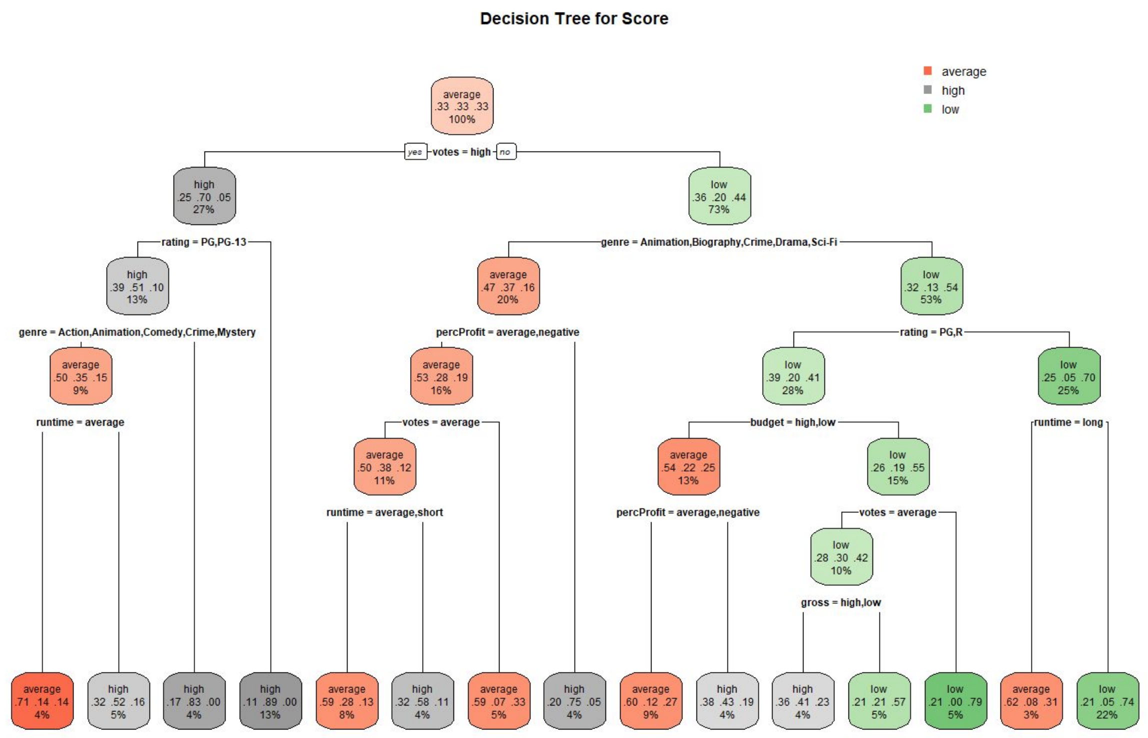
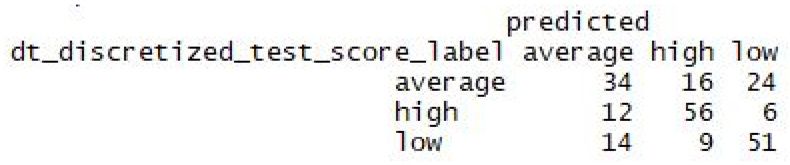


Figure 41. Decision Tree for Score

The minsplit was 40 and cp was set to 0. The decision tree for score contains 13 internal nodes. The original variables used in the decision tree are votes, rating, genre, percProf, runtime, budget, and gross.



```
                             predicted
dt_discretized_test_score_label average high low
                        average      34   16  24
                        high         12   56   6
                        low          14    9  51
```

Figure 42. Table of Results for Decision Tree

The decision tree had an overall accuracy of 64% when predicting score. The model accurately predicted low score 69%, average score 46%, and high scores 76% of the time. The model had the most difficulty with average scores.

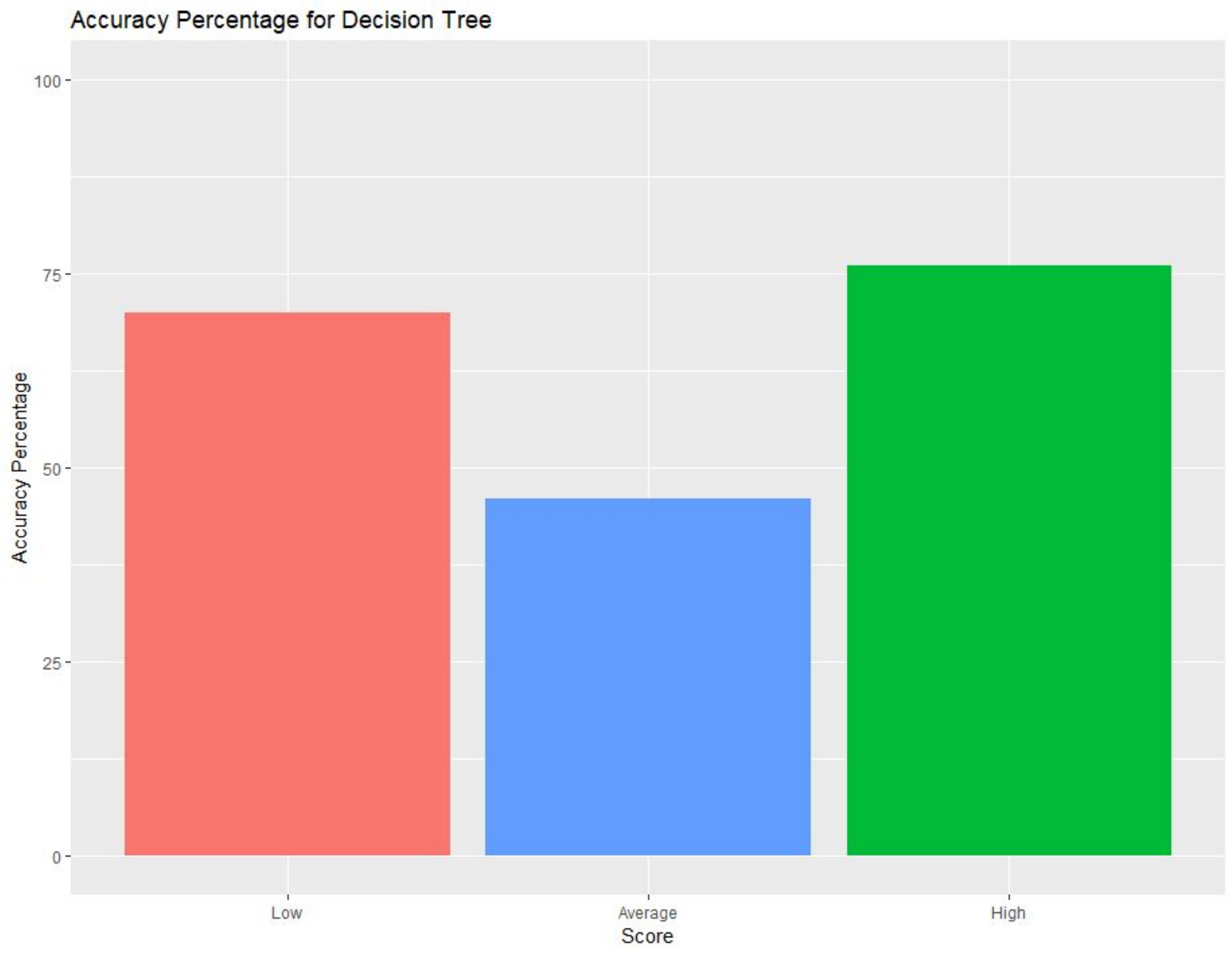
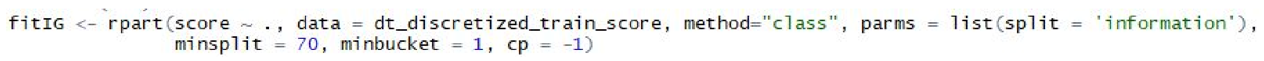


Figure 43. Visualization of Accuracy Percentage for Decision Tree

The second decision tree model included a parameter to split based on information gain. The minsplit was increased to 70, and cp was reduced to -1.



```
fitIG <- rpart(score ~ ., data = dt_discretized_train_score, method="class", parms = list(split = 'information'),
               minsplit = 70, minbucket = 1, cp = -1)
```

Figure 44. Code Snippet for Decision Tree with Information Gain

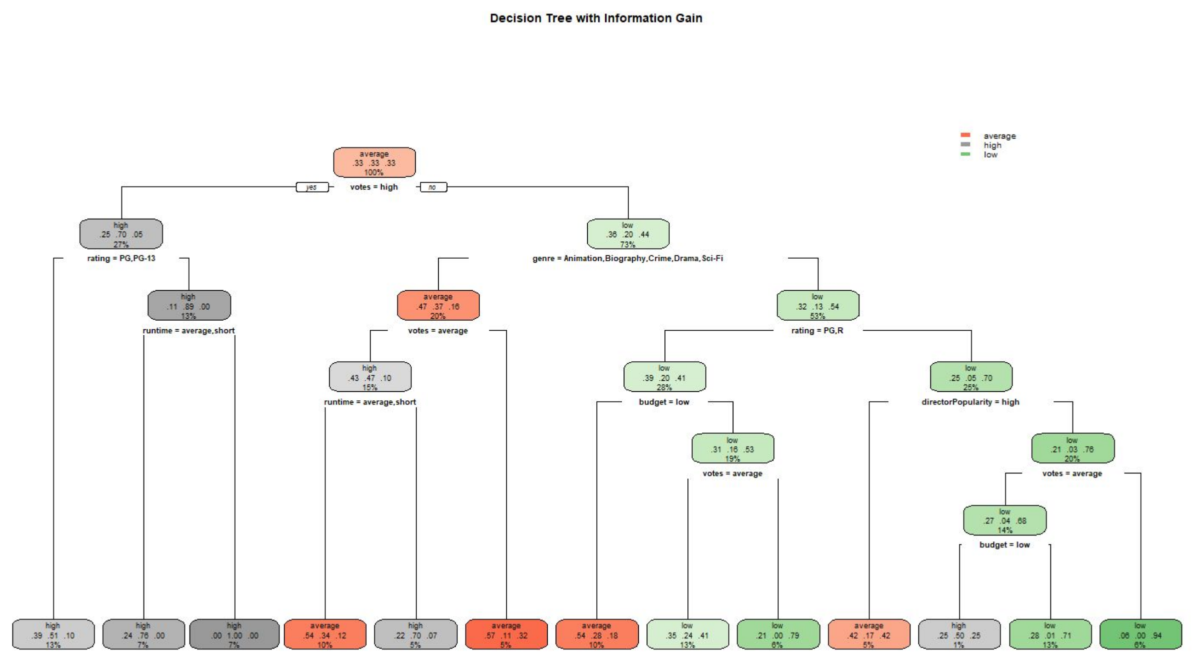


Figure 45. Visualization of Decision Tree with Information Gain

The decision tree with information gain has 11 internal nodes. The variables that the data was split by are votes, rating, genre, runtime, budget, and directorPopularity.

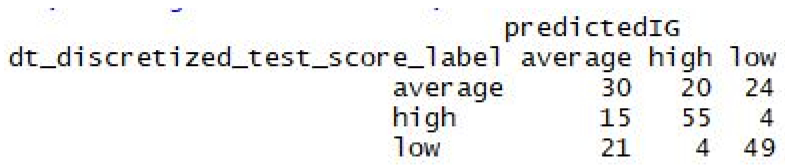

```
                           predictedIG
dt_discretized_test_score_label average high low
                        average      30   20  24
                        high         15   55   4
                        low          21    4  49
```

Figure 46. Table of Results for Decision Tree with Information Gain

The decision tree had an overall accuracy of 60% when predicting score. The model accurately predicted low score 66%, average score 41%, and high scores 74% of the time. The model had the most difficulty with average scores. Compared to the first decision tree, this model had a reduction in accuracy.

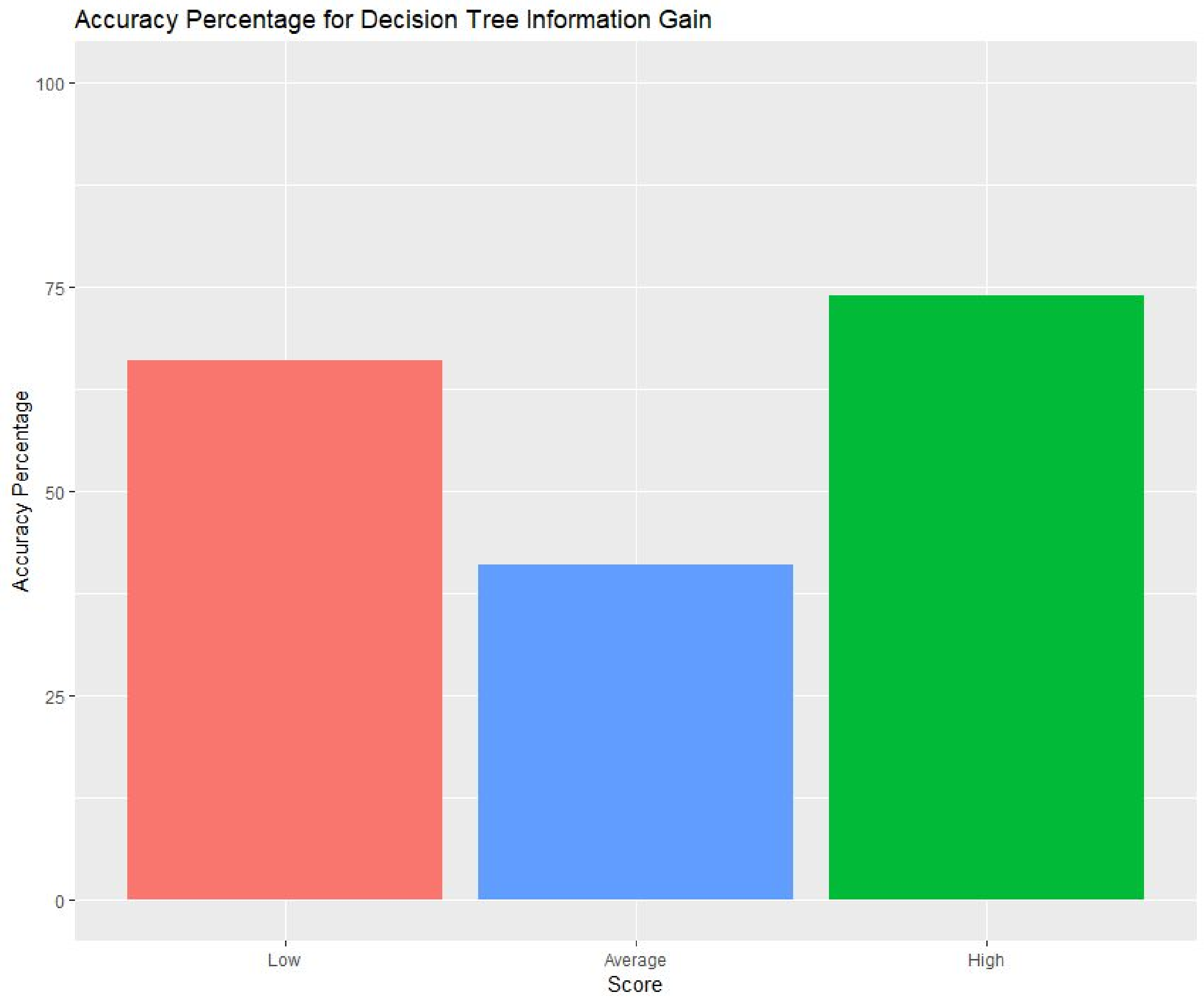


Figure 47. Visualization of Accuracy Percentage for Decision Tree with Information Gain

Comparison of Decision Tree Models for Score

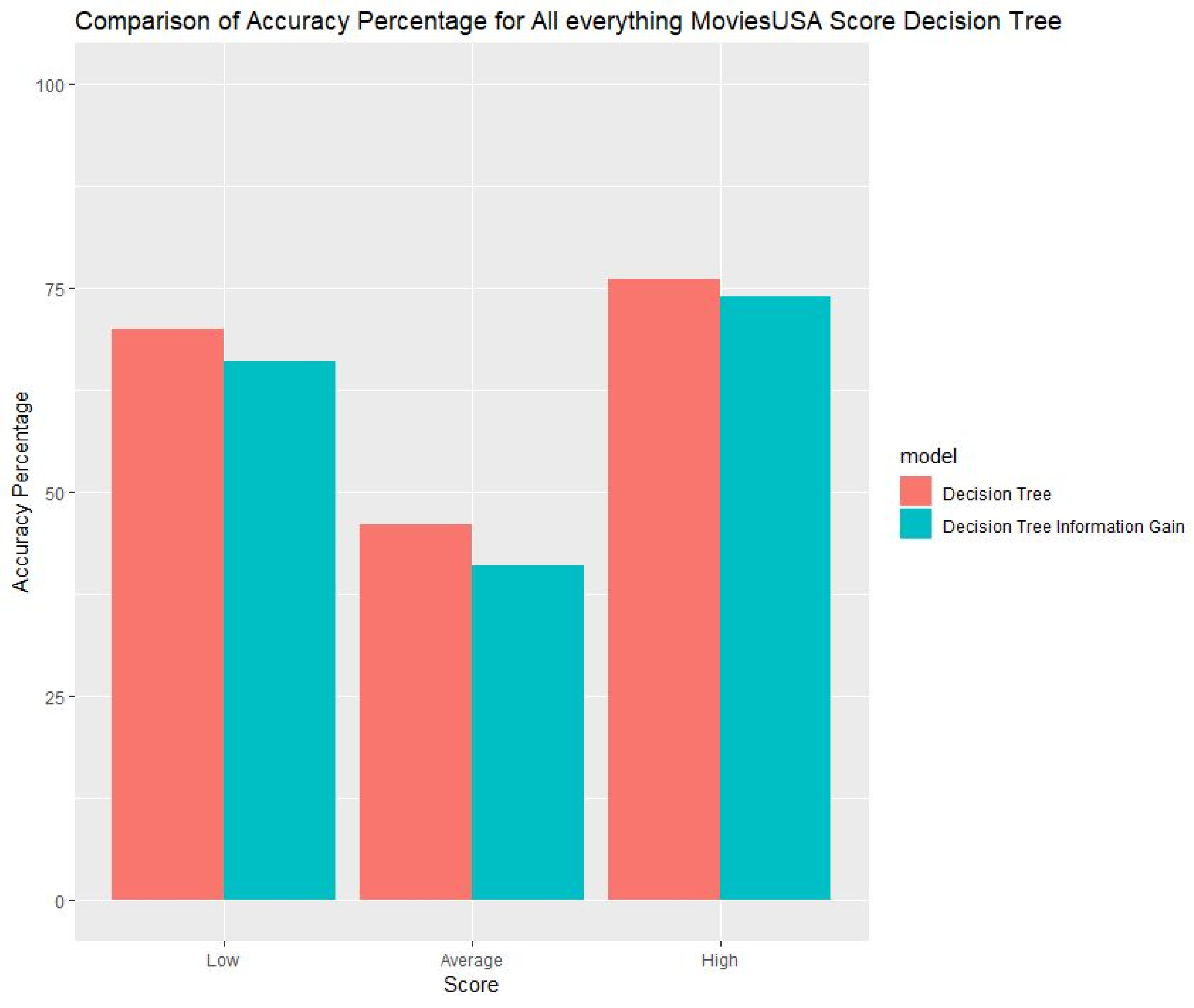
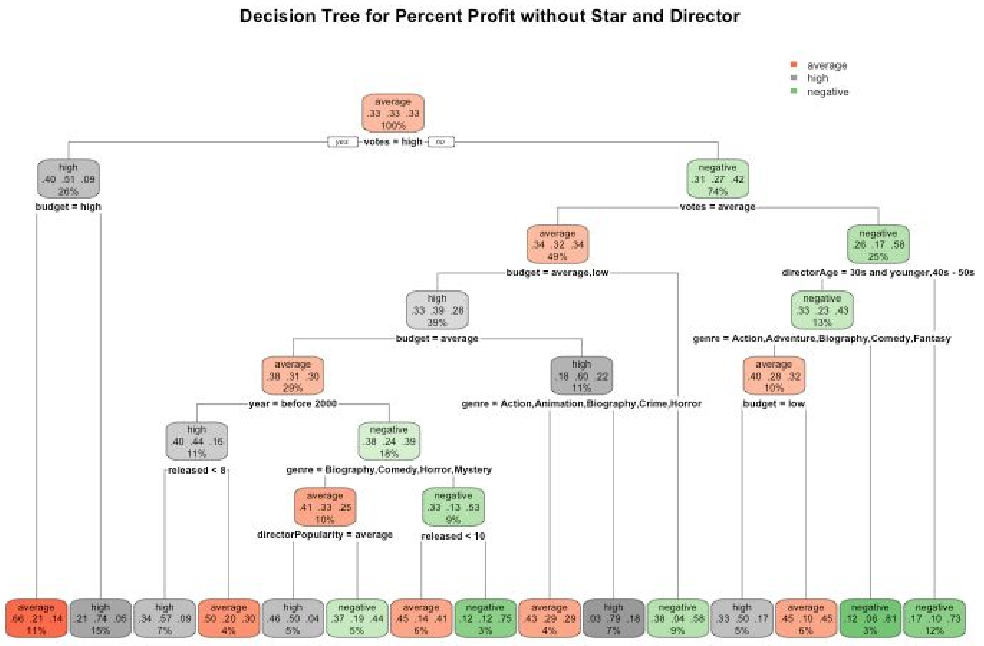


Figure 48. Visualization of the Comparison of Accuracy Percentage for All everything MoviesUSA Score Decision Tree

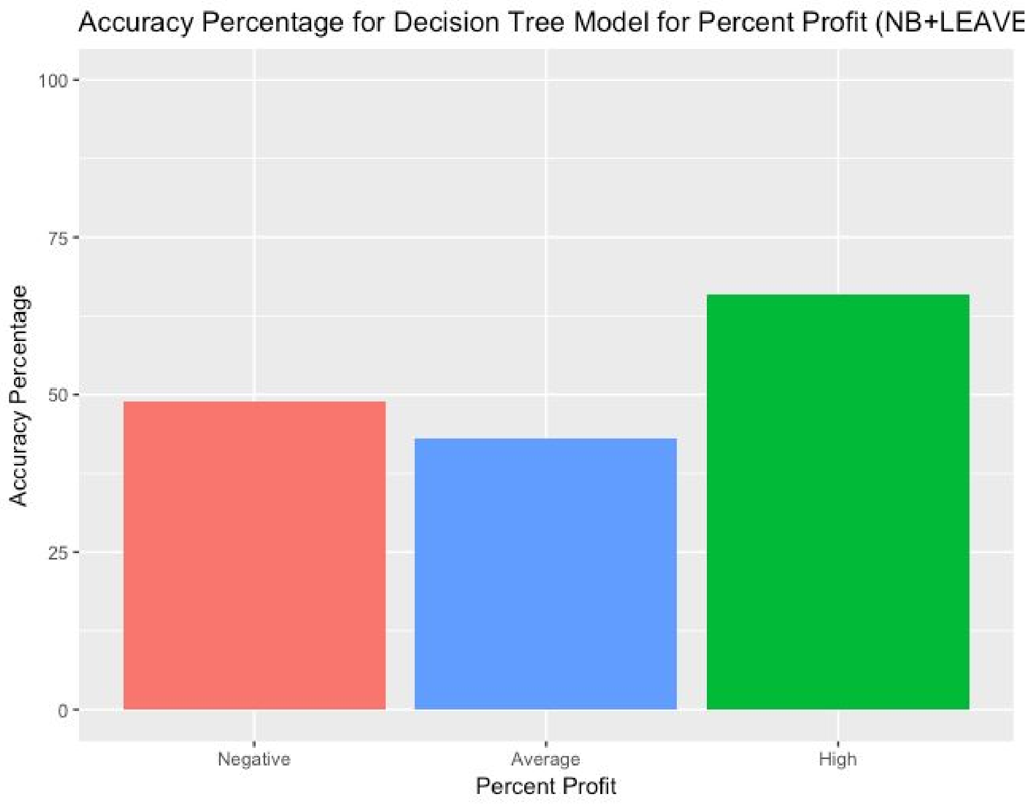
The decision tree without information gain was the most successful model when classifying low, average and high scores. Likewise it also had the highest overall accuracy with 64% compared to the information gain model of 60%.

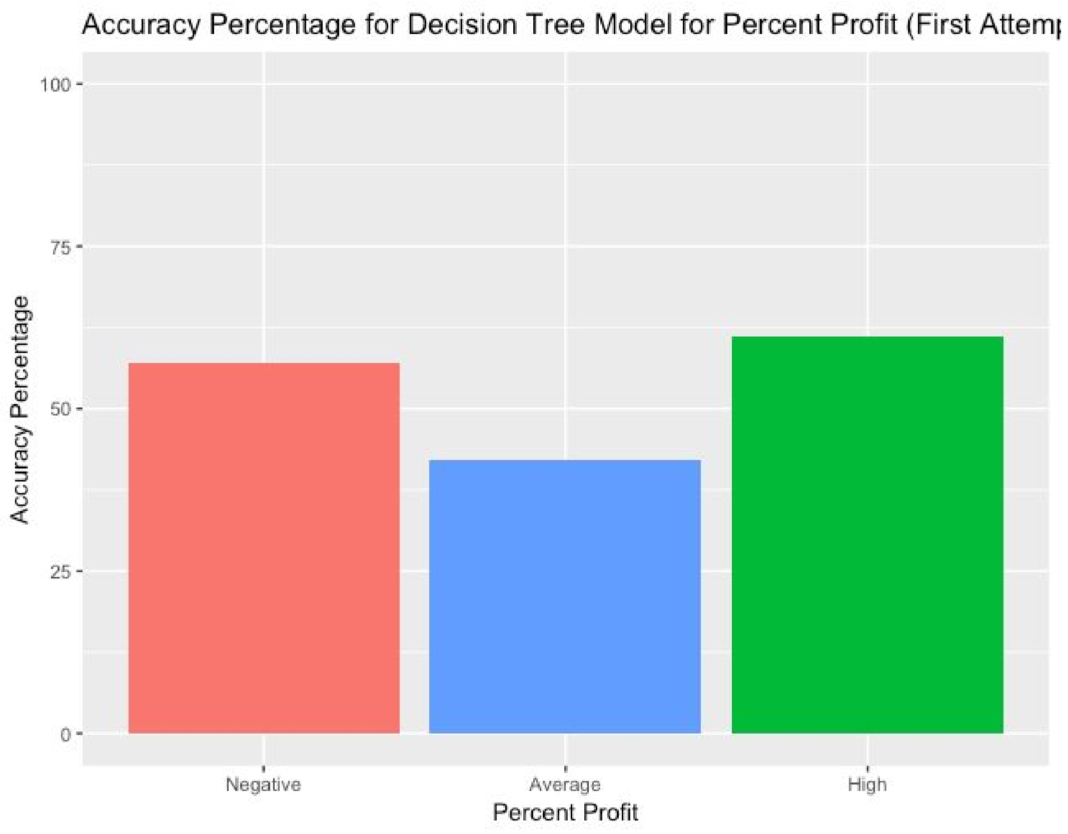
PERCENT PROFIT

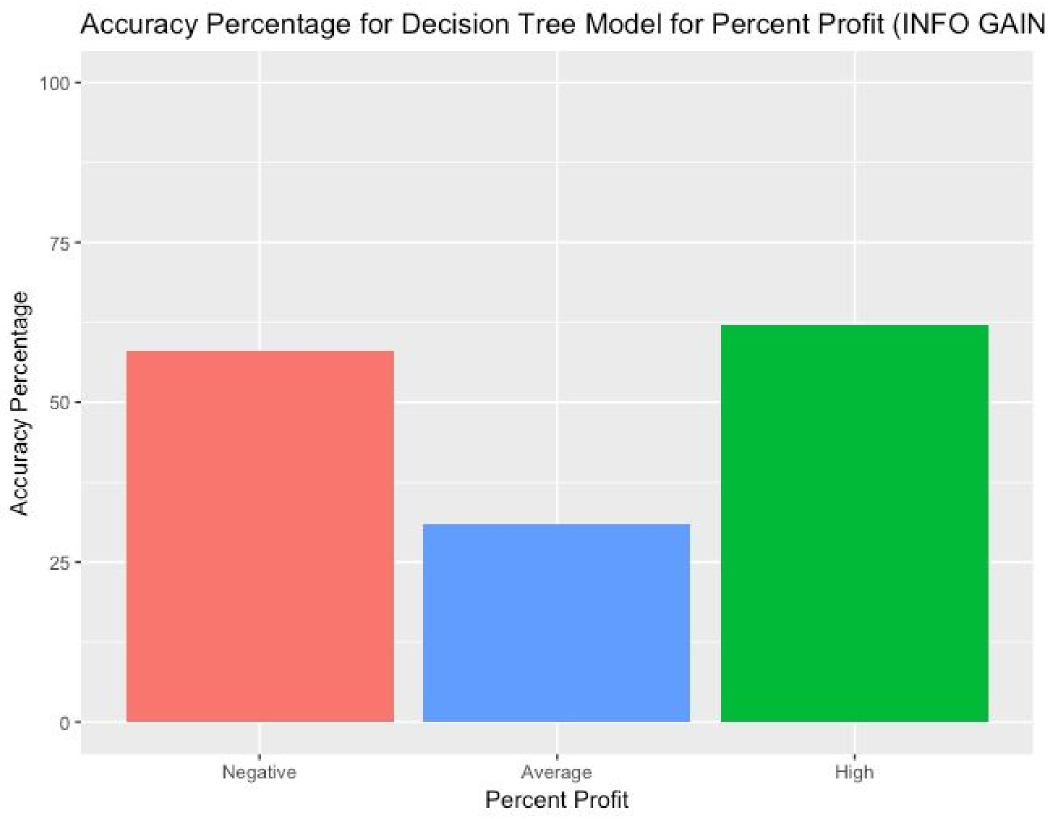
The same discretized dataset was used to predict percent profit. Similarly, the variables for title, director, star and released were removed from the original discretized dataset, as well as gross and profit. The data was split into testing and training data and fed into the decision tree model. Three models were created, the accuracy of each outlined below.

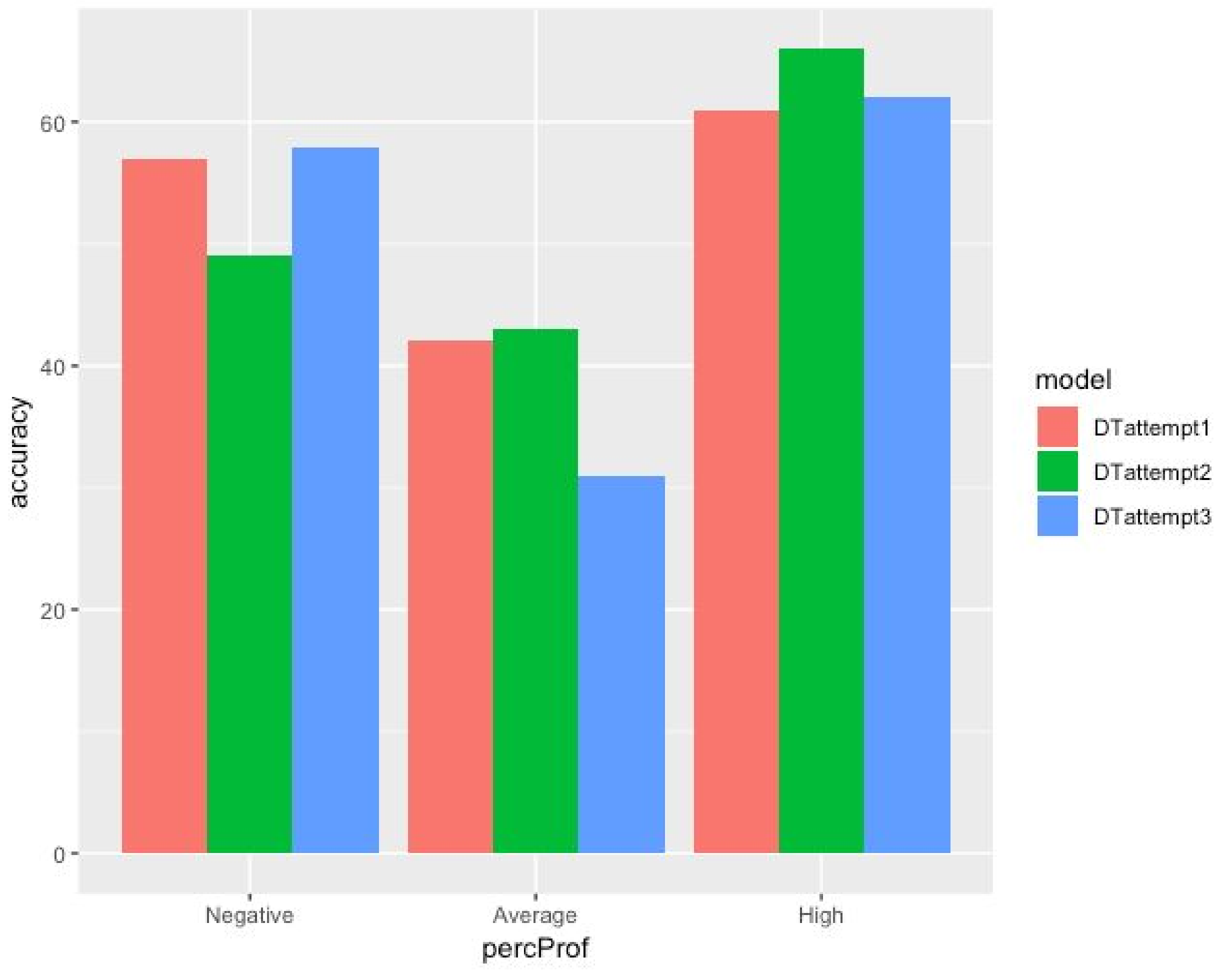


Figures 49-53.




Accuracy Percentage for Decision Tree Model for Percent Profit (First Attem
Accuracy Percentage
100
75
50
25
0
Negative
Average
High
Percent Profit


Accuracy Percentage for Decision Tree Model for Percent Profit (INFO GAIN
Accuracy Percentage
100
75
50
25
0
Negative
Average
High
Percent Profit



The results varied across models. Yet again, despite being discretized, normalized, prepped and re-prepped, the dataset, regardless the model within the model, did not predict percent profit with any level of statistically significant accuracy.

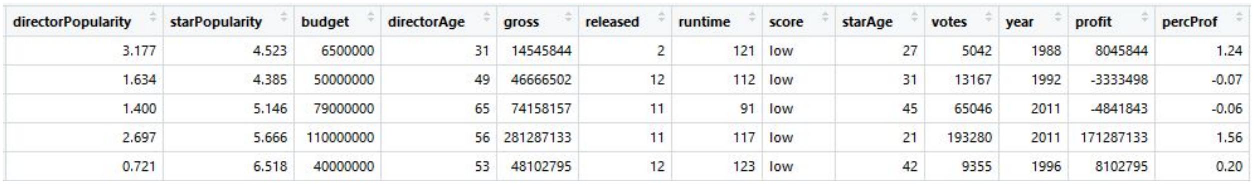
V. NAIVE BAYES

"I mean, it's sort of exciting, isn't it, breaking the rules?" — Hermione Granger

SCORE

Model 1

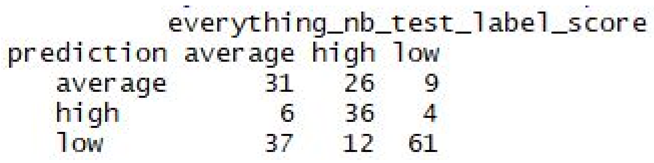
The first model utilized a dataset of all numeric variables with the score variable removed. The data frame contains 741 observations of 13 variables.



directorPopularity	starPopularity	budget	directorAge	gross	released	runtime	score	starAge	votes	year	profit	percProf
3.177	4.523	6500000	31	14545844	2	121	low	27	5042	1988	8045844	1.24
1.634	4.385	50000000	49	46666502	12	112	low	31	13167	1992	-3333498	-0.07
1.400	5.146	79000000	65	74158157	11	91	low	45	65046	2011	-4841843	-0.06
2.697	5.666	110000000	56	281287133	11	117	low	21	193280	2011	171287133	1.56
0.721	6.518	40000000	53	48102795	12	123	low	42	9355	1996	8102795	0.20

Figure 54. First 5 Rows of the Training Data Frame for Naive Bayes for Score

The data frame was then split into a training and test dataset. The training dataset contained 519 observations. There were 173 high scores, 173 average scores and 173 low scores. The testing dataset was comprised on 74 high scores, 74 average scores, and 74 low scores for a total of 222 observations. The score label was removed from the testing set.



```
           everything_nb_test_label_score
prediction average high low
   average      31   26   9
   high          6   36   4
   low          37   12  61
```

Figure 55. Table of Results for Naive Bayes Model 1

The first naive bayes model had an overall accuracy level of 58%. The model accurately predicted low scores 80%, average scores 42%, and high scores 49% of the time.

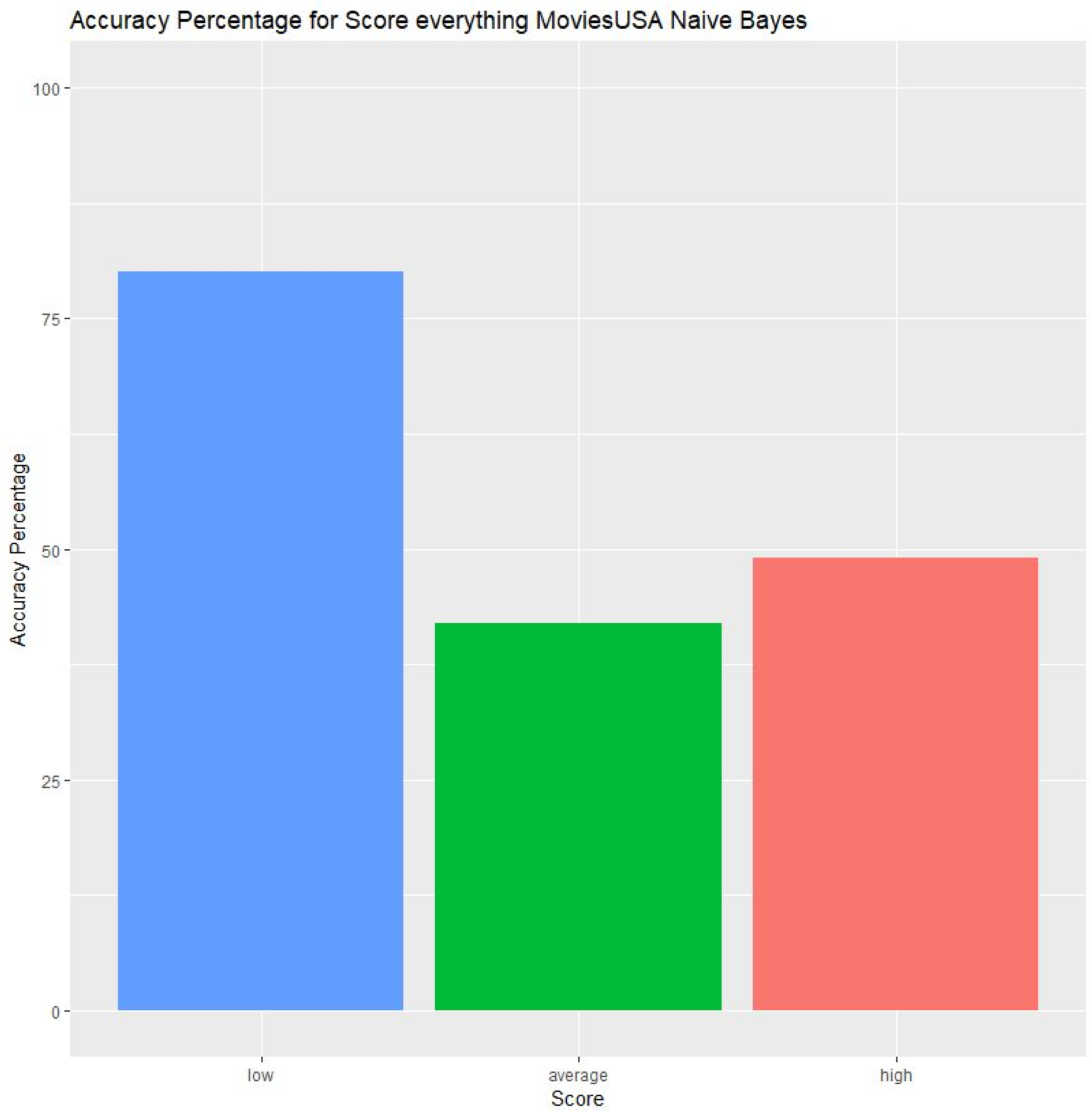
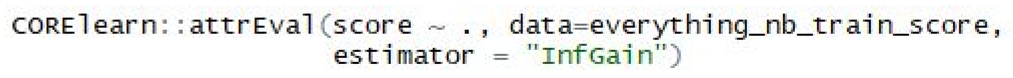


Figure 56. Visualization of Accuracy Percentage for Score everything MoviesUSA Naive Bayes

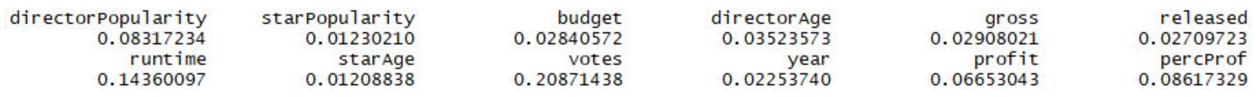
Model 2

The second model used a reduced data frame from attributes that CORElearn attribute eval with the information gain estimator deemed to be the most significant.



```
COREIearn::attrEval(score ~ ., data=everything_nb_train_score,
                    estimator = "InfGain")
```

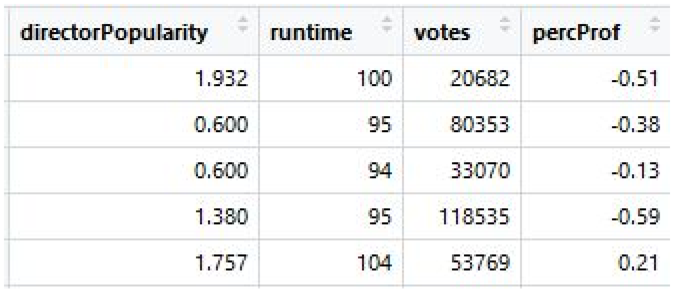
Figure 57. Code Snippet of COREIearn attribute eval



```
directorPopularity     starPopularity          budget        directorAge             gross          released
        0.08317234         0.01230210      0.02840572         0.03523573        0.02908021        0.02709723
           runtime            starAge           votes               year            profit          percProf
        0.14360097         0.01208838      0.20871438         0.02253740        0.06653043        0.08617329
```

Figure 58. Output of COREIearn attribute evaluation

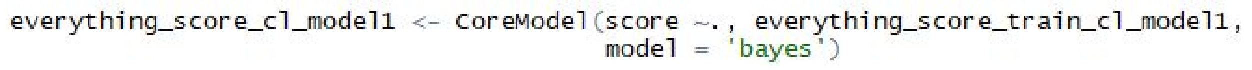
Core learn with information gain determined that the attributes that yield the greatest amount of information gain are runtime, percProf, and directorPopularity. The dataset was reduced to include only those 3 attributes.



directorPopularity	runtime	votes	percProf
1.932	100	20682	-0.51
0.600	95	80353	-0.38
0.600	94	33070	-0.13
1.380	95	118535	-0.59
1.757	104	53769	0.21

Figure 59. First 5 Rows of the Training Data Frame for Naive Bayes for Score Model 2

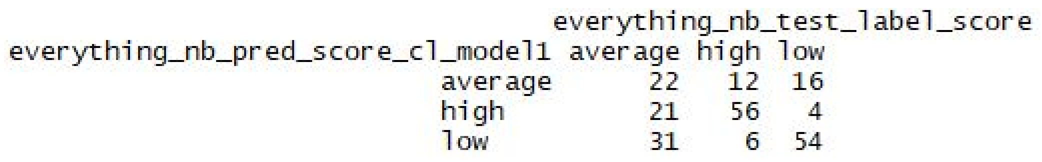
The second attempt at naive bayes was run with the CoreModel function and specified the model as naive bayes.



```
everything_score_cl_model1 <- CoreModel(score ~., everything_score_train_cl_model1,
                                        model = 'bayes')
```

Figure 60. Code Snippet to Run Naive Bayes with the CoreModel function from the COREIearn Package

The CoreModel Naive Bayes model with the reduced data frame had an overall accuracy of 59%.



```
                                   everything_nb_test_label_score
everything_nb_pred_score_cl_model1 average high low
                           average      22   12  16
                           high         21   56   4
                           low          31    6  54
```

Figure 61. Table of Results for COREIearn NB with DF Reduced by Information Gain

The model accurately predicted low scores 73%, average scores 29% and high scores 73% of the time.

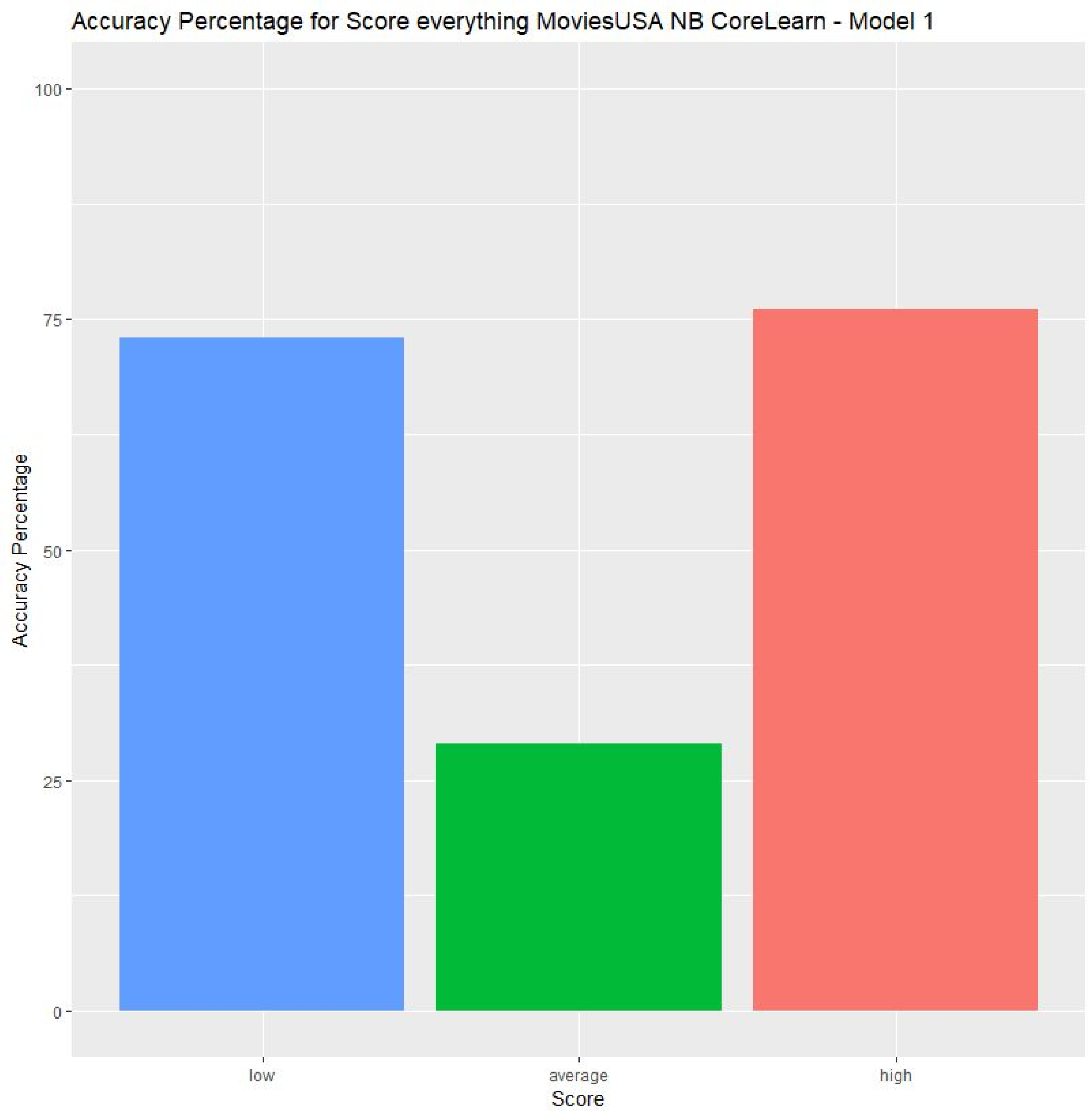
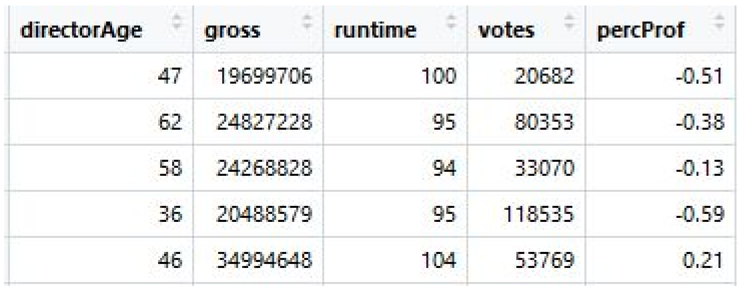


Figure 62. Visualization of Accuracy Percentage for Score Naive Bayes CoreLean Model 1

Model 3

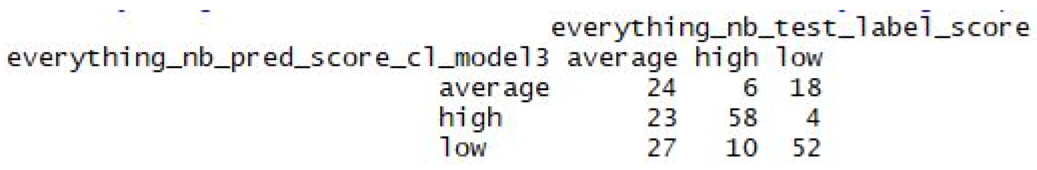
The third model used a data frame that was reduced in a similar way as model two, however instead of using information gain as the estimator gainRatio was used. COREIearn deemed that votes, runtime, gross, directorAge, and percProf were the best attributes to use.



directorAge	gross	runtime	votes	percProf
47	19699706	100	20682	-0.51
62	24827228	95	80353	-0.38
58	24268828	94	33070	-0.13
36	20488579	95	118535	-0.59
46	34994648	104	53769	0.21

Figure 63. First 5 Rows of the Testing Data Frame for Naive Bayes for Score Model 3

The third model was run using the same method as the second model. The CoreModel Naive Bayes had an overall accuracy of 60%.



```
                                  everything_nb_test_label_score
everything_nb_pred_score_cl_model3 average high low
                           average      24    6  18
                           high         23   58   4
                           low          27   10  52
```

Figure 64. Table of Results for cl model 3

The model accurately predicted low scores 70%, average scores 32% and high scores 78% of the time.

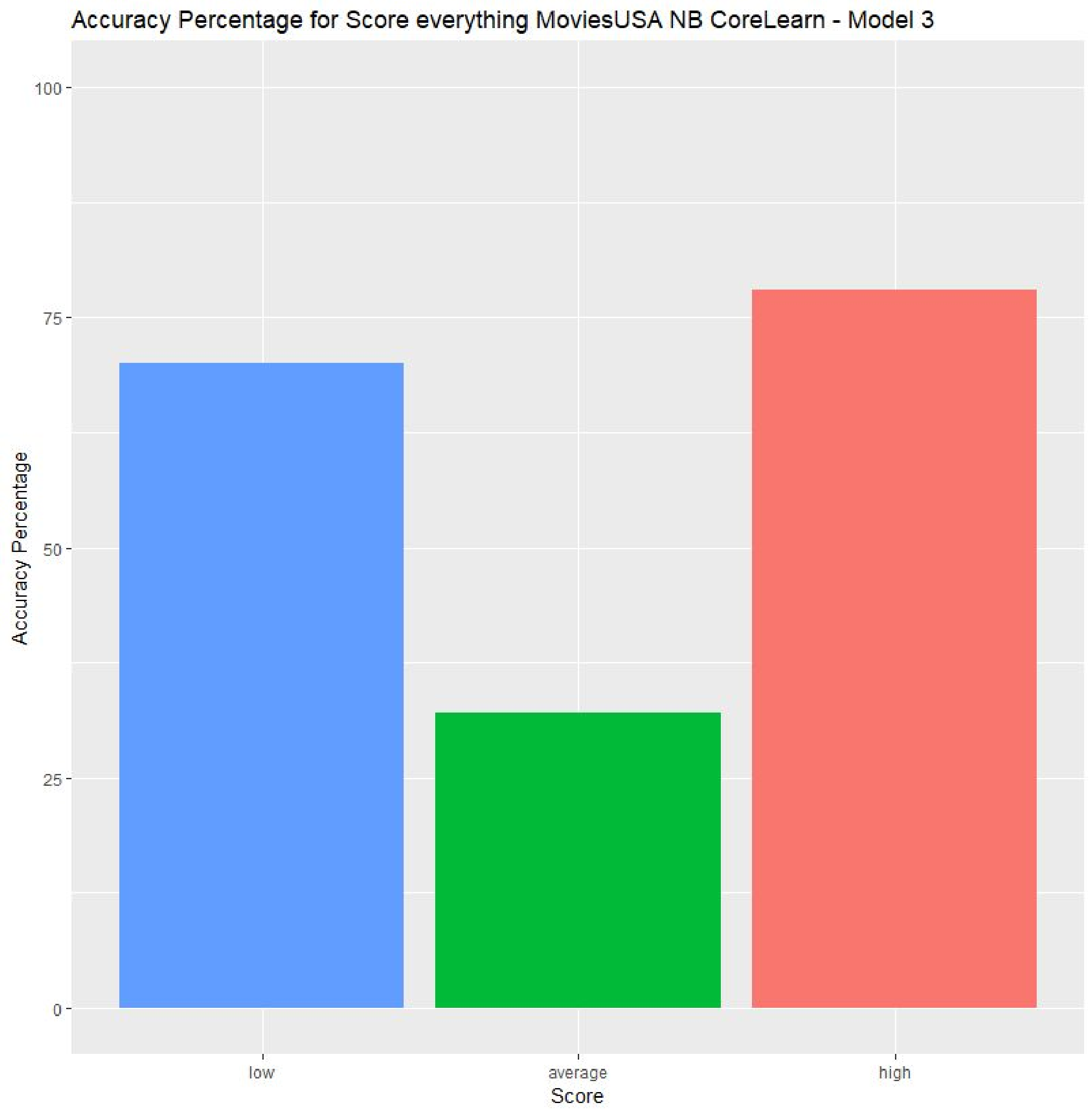


Figure 65. Visualization of Accuracy Percentage for NB Score CoreLean - Model 3

Comparison of Naive Bayes Models for Score

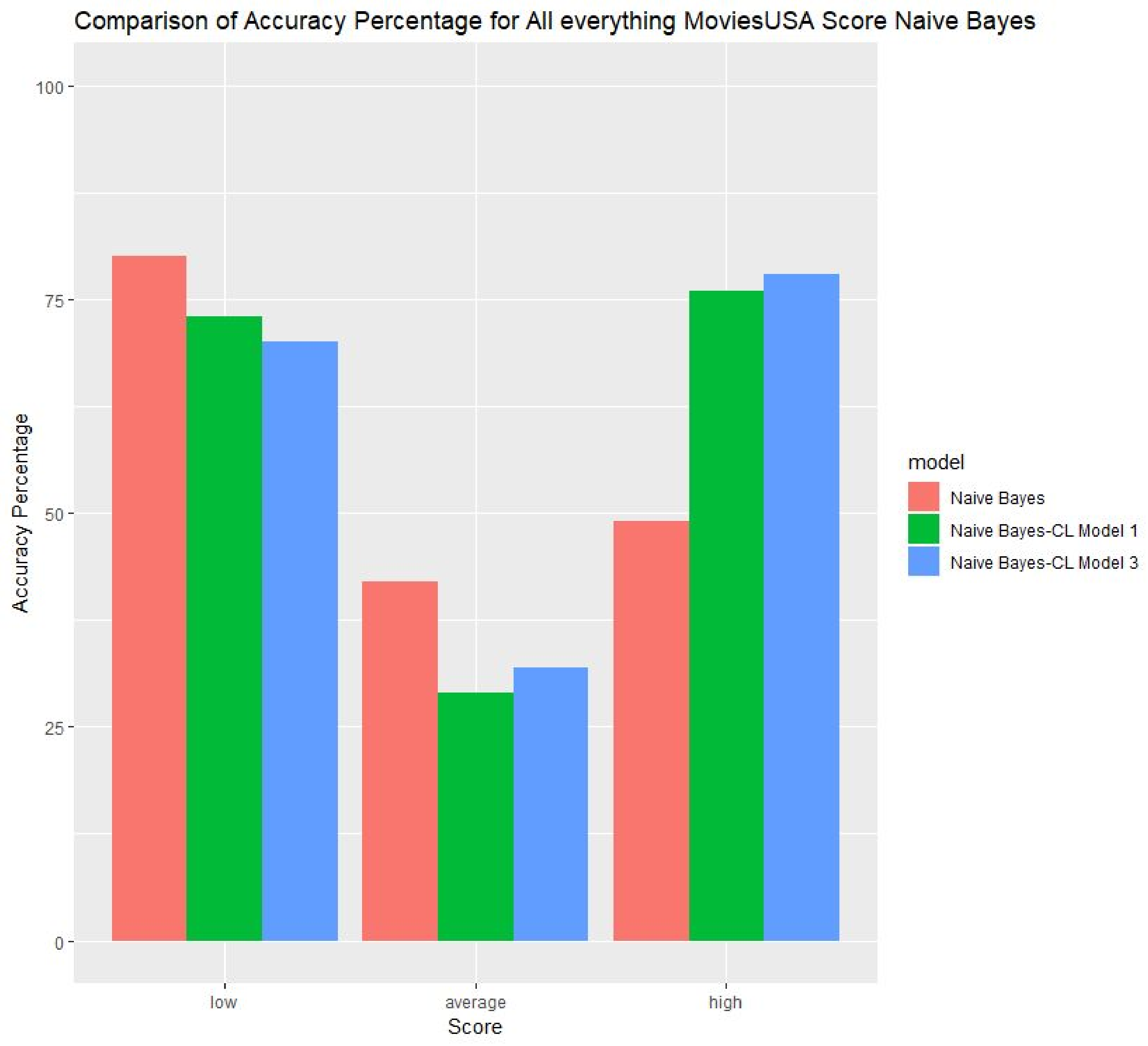


Figure 66. Visualization of Comparison of Accuracy Percentage for All Everything MoviesUSA Score Naive Bayes

The three models had their own strengths and weaknesses when classifying score. All models had a difficult time classifying average scores. The original model excelled classifying low scores, but suffered when classifying both average and high scores. The Cl Model 1 nicely classified low and high scores, but had the worst classification accuracy for average scores. Cl Model 3 had the best accuracy when classifying high scores, but the worst accuracy when classifying low scores.

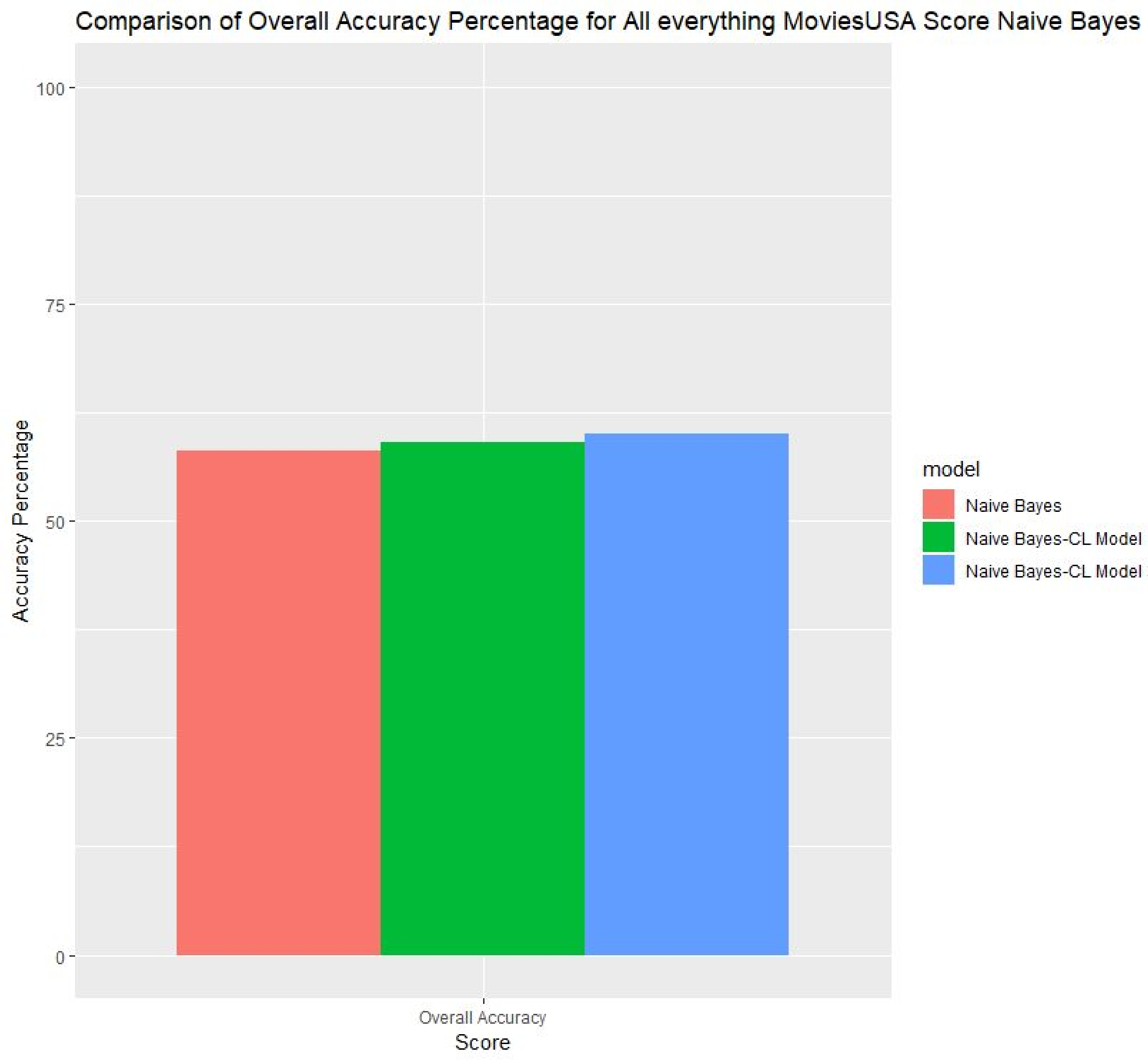


Figure 67. Comparison of Overall Accuracy Percentage for All everything MoviesUSA Score Naive Bayes

The three models had similar overall accuracy percentages at 58%, 59% and 60%. It is important to decide which accuracy levels are most important in the dataset to ultimately decide which model performed the best for Naive Bayes.

SCORE PLUS (Explaining the 81%)

After getting discouraged one too many times, a certain researcher set out to poke and prod the data a slightly different way than before. Instead of using singular numbers, the researcher decided to aggregate and average both the star and the director's score history.

This, combined with aggressive discretization, lead to the giant, intentionally provocative, headline that will encourage business moguls and universities to invest more heavily in the truly, *truely* disruptive research.

```
## ====================================
## AGGREGATE & MATH THE DATA
## ====================================
## FOR STARS *************************
## & SCORES ***************************
starAvgScore <- aggregate(df$score, by=list(star=df$star), FUN=mean)
starMaxScore <- aggregate(df$score, by=list(star=df$star), FUN=max)
starMinScore <- aggregate(df$score, by=list(star=df$star), FUN=min)
starFrequency <- aggregate(df$score, by=list(star=df$star), FUN=length)
starPower <- cbind(starAvgScore, starMaxScore[,2], starMinScore[,2],
starFrequency[,2])

## & PercPROF *************************
starAvgPercProf <- aggregate(df$percProf, by=list(star=df$star), FUN=mean)
starMaxPercProf <- aggregate(df$percProf, by=list(star=df$star), FUN=max)
starMinPercProf <- aggregate(df$percProf, by=list(star=df$star), FUN=min)

## combining & renaming
starPower <- cbind(starAvgScore, starMaxScore[,2], starMinScore[,2],
starFrequency[,2],
                   starAvgPercProf[,2], starMaxPercProf[,2],
starMinPercProf[,2])
colnames(starPower) <- c('star', 'starScoreAvg',
'starScoreMax','starScoreMin', 'starFrequency',
                         'starPPAvg', 'starPPMax', 'starPPMin')

## FOR DIRECTORS **********************
## & SCORES ***************************
directorAvgScore <- aggregate(df$score, by=list(director=df$director),
FUN=mean)
directorMaxScore <- aggregate(df$score, by=list(director=df$director),
FUN=max)
directorMinScore <- aggregate(df$score, by=list(director=df$director),
FUN=min)
directorFrequency <- aggregate(df$score, by=list(director=df$director),
FUN=length)
directorPower <- cbind(directorAvgScore, directorMaxScore[,2],
directorMinScore[,2], directorFrequency[,2])
```

```
## & PercPROF ***************************
directorAvgPercProf <- aggregate(df$percProf,
by=list(director=df$director), FUN=mean)
directorMaxPercProf <- aggregate(df$percProf,
by=list(director=df$director), FUN=max)
directorMinPercProf <- aggregate(df$percProf,
by=list(director=df$director), FUN=min)

## combining & renaming
directorPower <- cbind(directorAvgScore, directorMaxScore[,2],
directorMinScore[,2], directorFrequency[,2],
                       directorAvgPercProf[,2], directorMaxPercProf[,2],
directorMinPercProf[,2])
colnames(directorPower) <- cbind('director', 'directorScoreAvg',
'directorScoreMax', 'directorScoreMin', 'directorFrequency',
                                 'directorPPAvg', 'directorPPMax',
'directorPPMin')
```

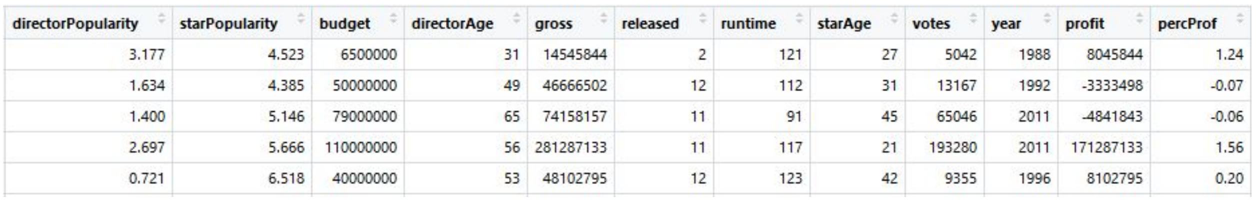
The aggressive mathing and discretization resulted in the most promising accuracy percentage yet. More testing is definitely needed but this serves to reinforce the importance of strong numeric data points in a dataset when using such models.

VI. KNN

SCORE

Model 1

The first model utilized a dataset of all numeric variables with the score variable removed. The data frame contains 741 observations of 13 variables.

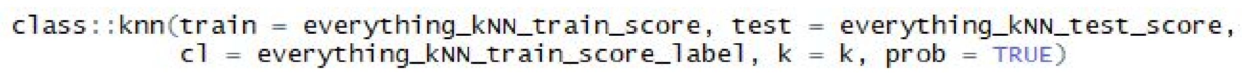


directorPopularity	starPopularity	budget	directorAge	gross	released	runtime	starAge	votes	year	profit	percProf
3.177	4.523	6500000	31	14545844	2	121	27	5042	1988	8045844	1.24
1.634	4.385	50000000	49	46666502	12	112	31	13167	1992	-3333498	-0.07
1.400	5.146	79000000	65	74158157	11	91	45	65046	2011	-4841843	-0.06
2.697	5.666	110000000	56	281287133	11	117	21	193280	2011	171287133	1.56
0.721	6.518	40000000	53	48102795	12	123	42	9355	1996	8102795	0.20

Figure 68. First 5 Rows of the Training Data Frame for kNN for Score

The data frame was then split into a training and test dataset. The training dataset contained 519 observations. There were 173 high scores, 173 average scores and 173 low scores. The testing dataset was comprised on 74 high scores, 74 average scores, and 74 low scores for a total of 222 observations. The score label was removed from both the training and testing set.

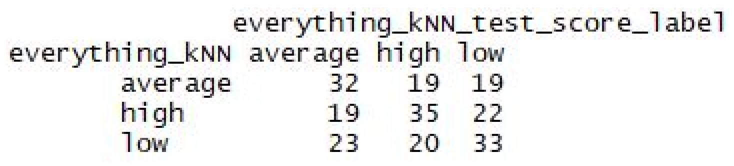
The k for the first model was determined by taking the square root of the total number of entries in both the testing and training data frames. This set the k at 27.



```
class::knn(train = everything_kNN_train_score, test = everything_kNN_test_score,
           cl = everything_kNN_train_score_label, k = k, prob = TRUE)
```

Figure 69. Code Snippet for kNN model 1

Knn requires the training data, the testing data, the training label, k and probability. The overall accuracy of the model was 45%.



```
               everything_kNN_test_score_label
everything_kNN average high low
       average      32   19  19
       high         19   35  22
       low          23   20  33
```

Figure 70. Table of Results for k = 27 Model 1

The model accurately predicted low scores 45%, average scores 43%, and high scores 47% of the time. The model does not accurately predict score and is significantly less accurate than the previous models.

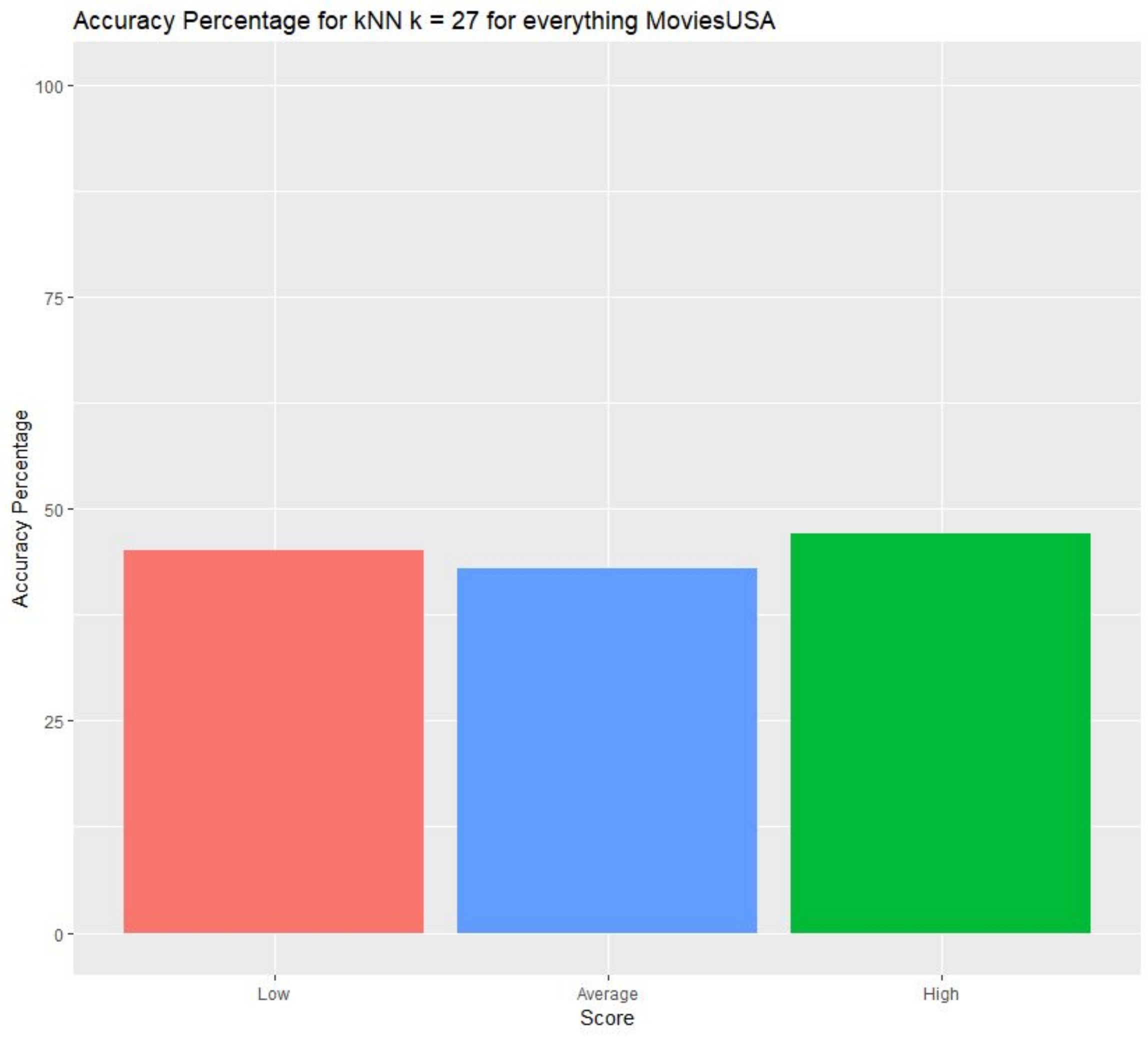
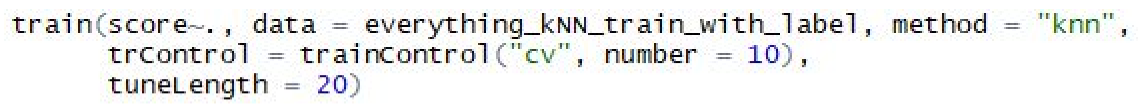


Figure 71. Visualization of Accuracy Percentage for kNN k = 27

To find the best k value for the dataset kNN was run using the caret package. The model included 20 different possible k values and train control set up a 10-fold cross validation. The model required the training data frame with the label and the method to run.



```
train(score~., data = everything_kNN_train_with_label, method = "knn",
      trControl = trainControl("cv", number = 10),
      tuneLength = 20)
```

Figure 72. Code Snippet to Determine the Best K-Value

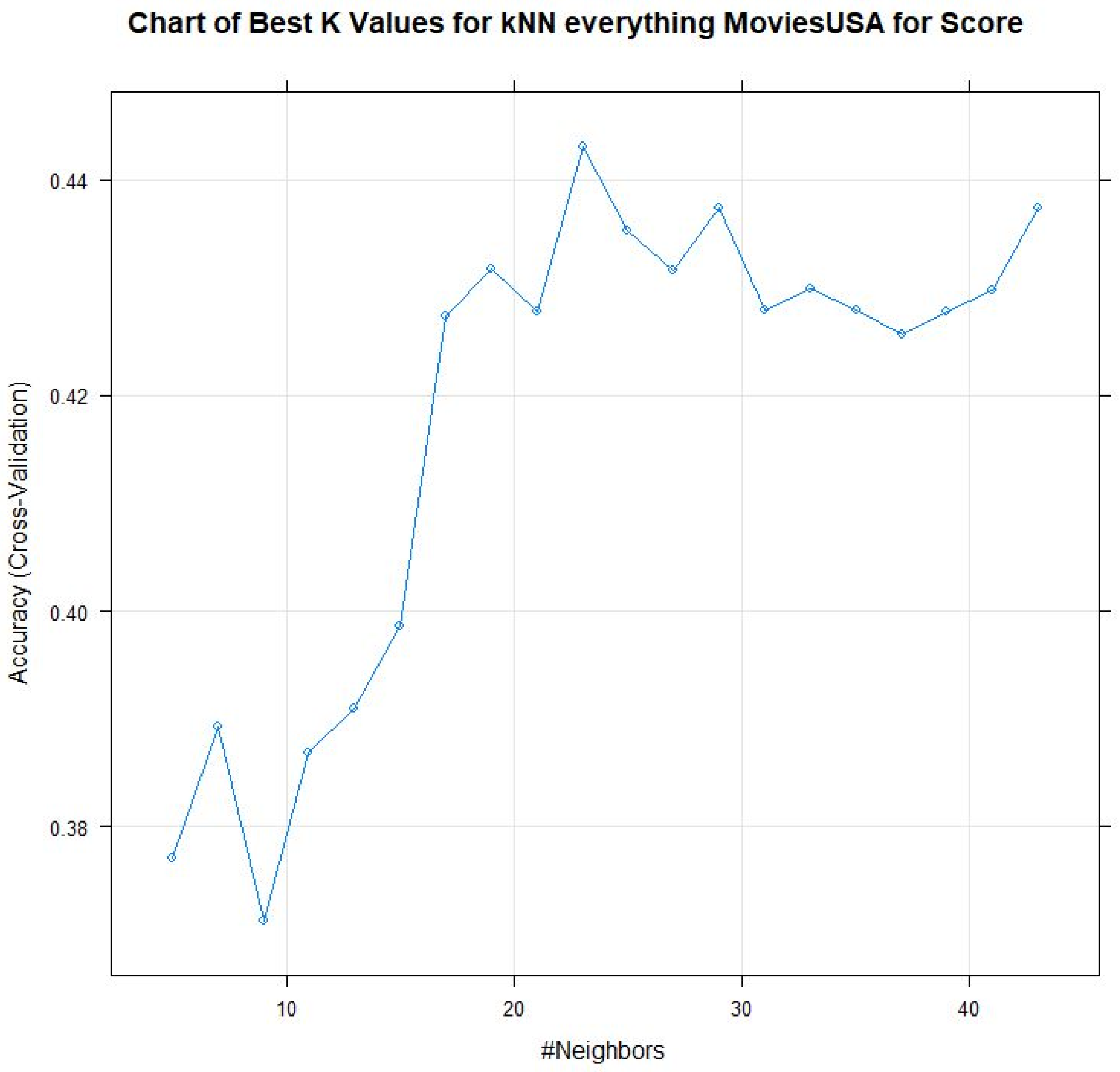
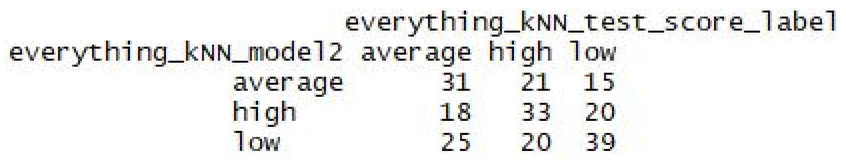


Figure 73. Chart of Best K Values for kNN everythingMoviesUSA for Score

The graph and running the following line of code "everything_model_kNN$bestTune" determined that the best k value is 23.

The second attempt for kNN used a k of 23.



```
                          everything_kNN_test_score_label
everything_kNN_model2 average high low
              average      31   21  15
              high         18   33  20
              low          25   20  39
```

Figure 74. Table of Results for kNN k = 23 for everything_MoviesUSA

The change in k, only increased the overall accuracy of the model to 46%. The model accurately predicted low scores 53%, average scores 42%, and high scores 45% of the time.

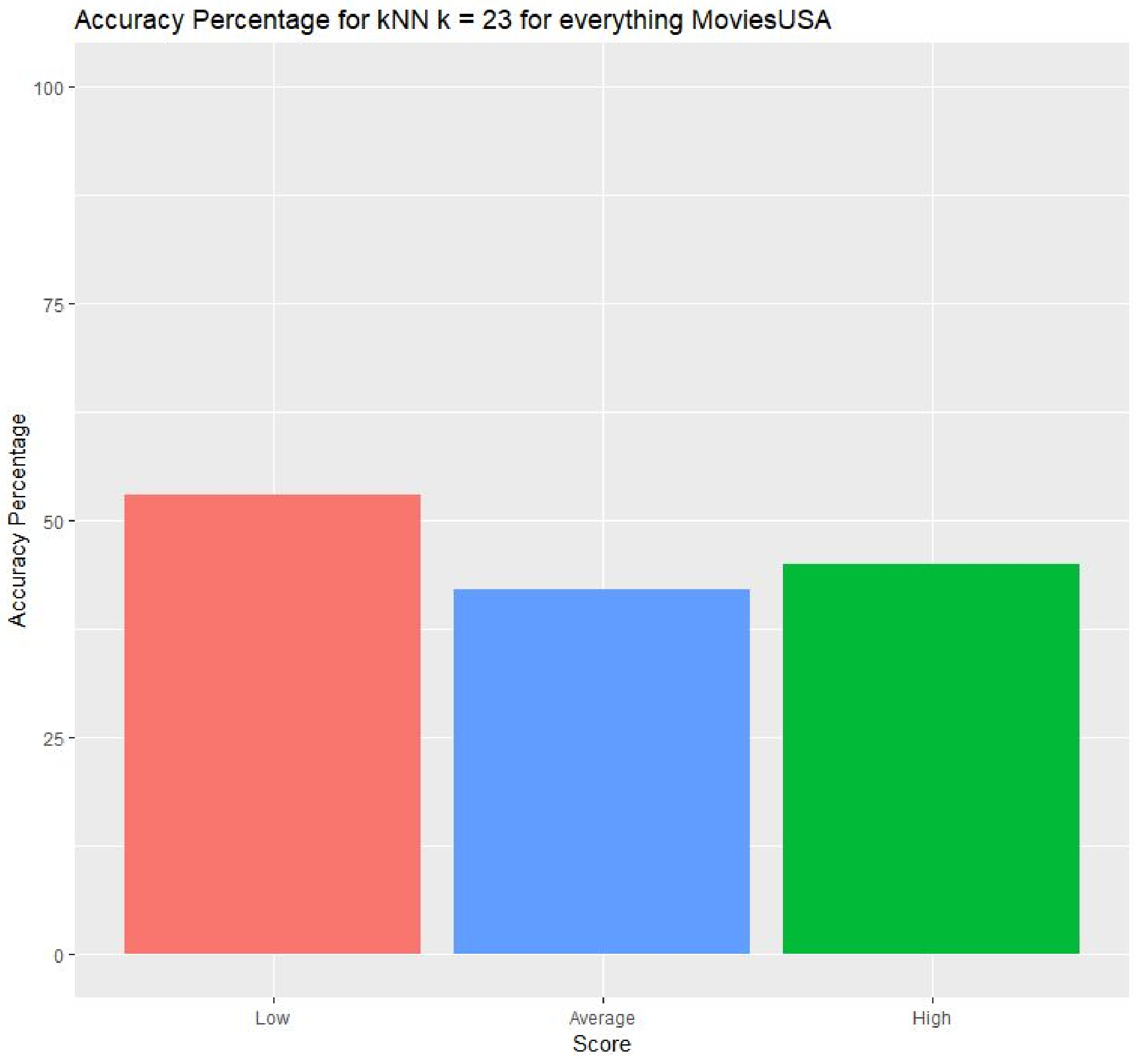
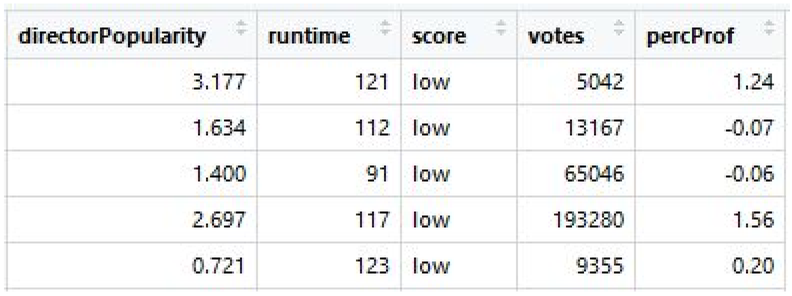


Figure 75. Visualization of Accuracy Percentage for kNN k = 23 for everything_MoviesUSA Score

Model 2

In an attempt to improve the results for kNN, the data frame was reduced to include only votes, runtime, percProf, and directorPopularity.



directorPopularity	runtime	score	votes	percProf
3.177	121	low	5042	1.24
1.634	112	low	13167	-0.07
1.400	91	low	65046	-0.06
2.697	117	low	193280	1.56
0.721	123	low	9355	0.20

Figure 76. First 5 Rows of the Reduced Data Frame for Model 2

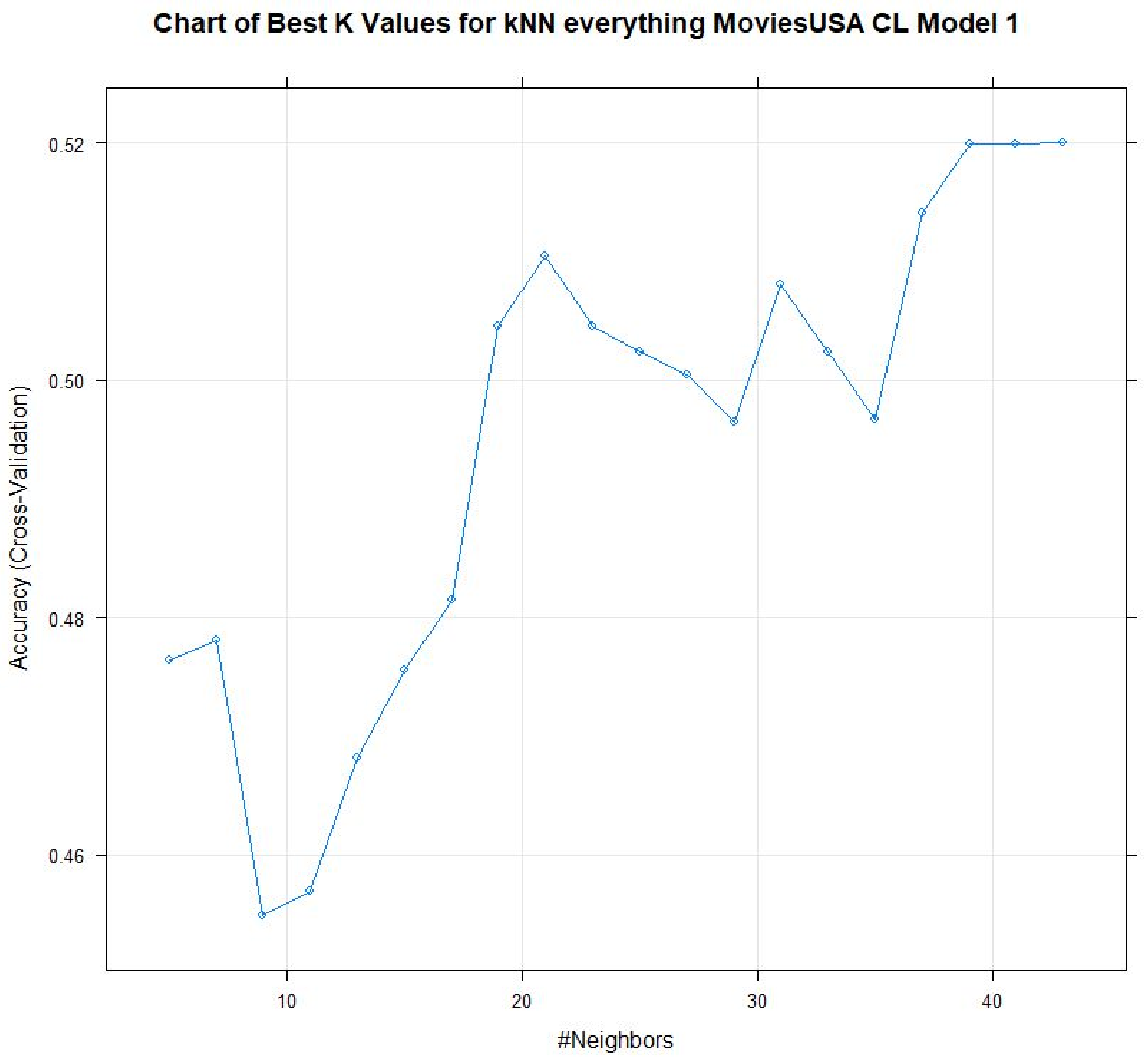
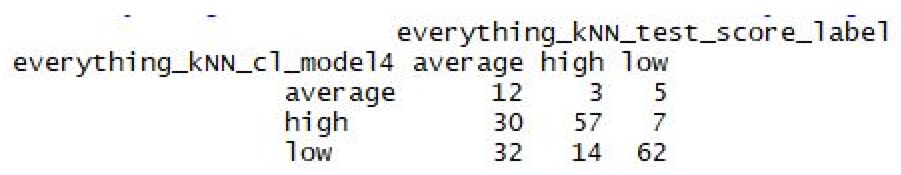


Figure 77. Chart of Best K Values for kNN everything_MoviesUSA CL Model 1 - Reduced

The best k value was determined to be 43. The k was set for 43 for this model.



```
                        everything_kNN_test_score_label
everything_kNN_cl_model4 average high low
                 average      12    3   5
                 high         30   57   7
                 low          32   14  62
```

Figure 78. Table of Results for k = 43 for CL Model 1 Reduced DF

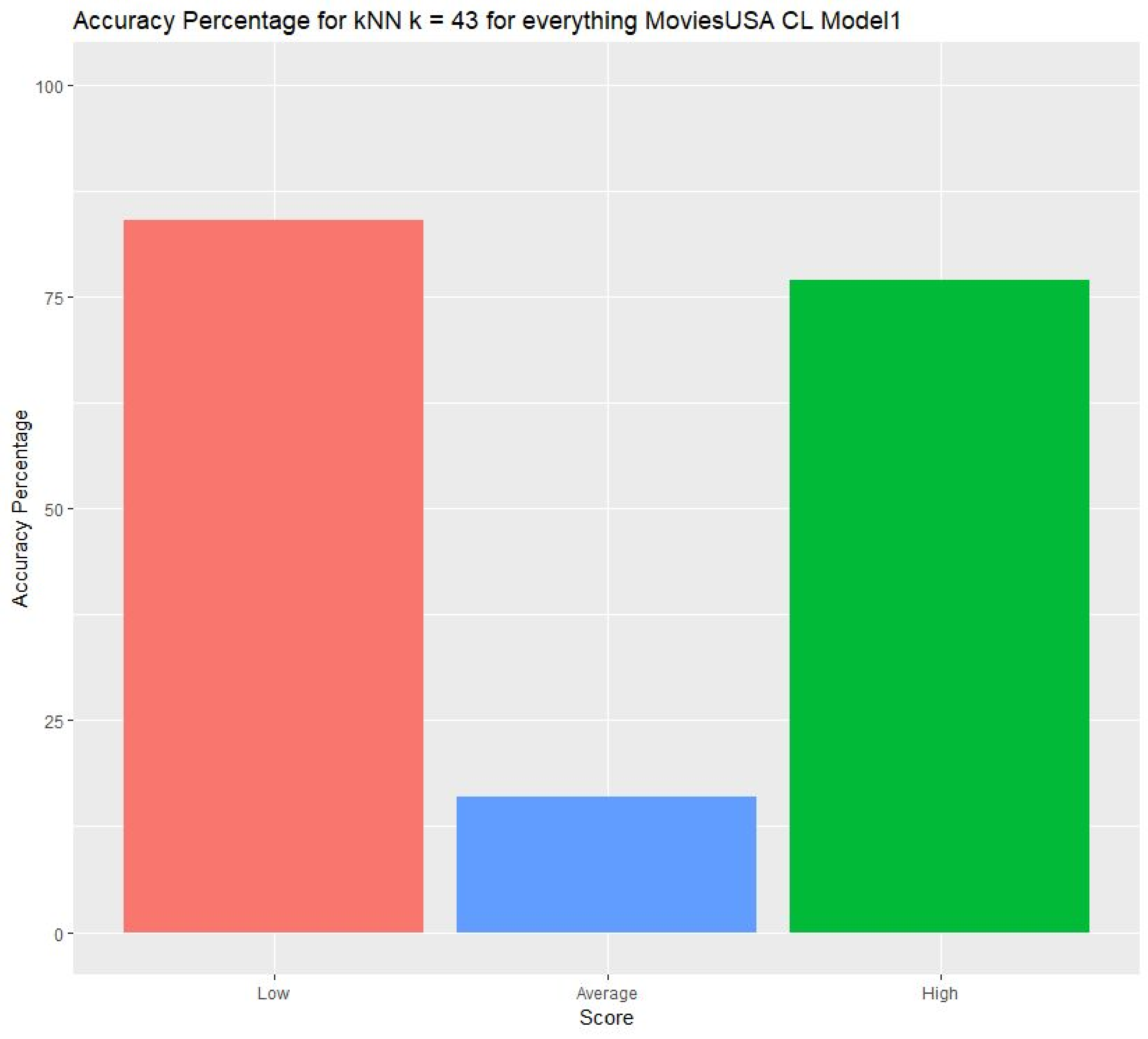


Figure 79. Visualization of Accuracy Percentage for kNN k = 43 for everything MoviesUSA CL Model 1

The model did a nice job of classifying low and high scores, however had a difficult time classifying average scores with an accuracy of 16%.

Two other models were run with different attributes included in the data frame. However, both of those models had an accuracy rate of 43% or less and will not be discussed in the paper.

Comparison of All Models for kNN Score

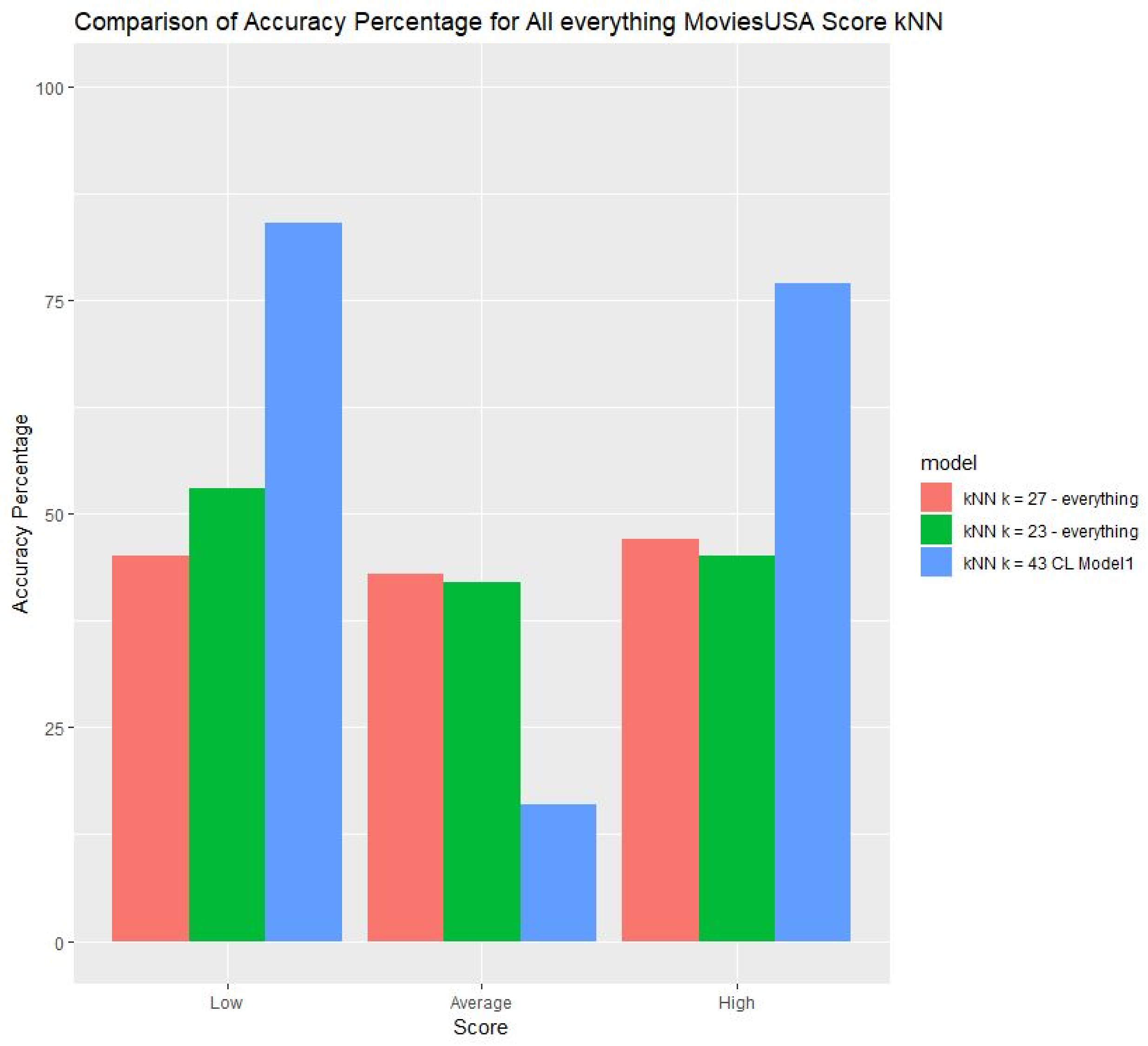
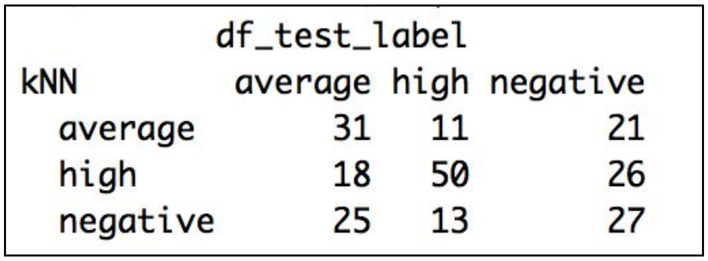


Figure 80. Visualization of Accuracy Percentage of All everything MoviesUSA Score kNN

The third model, cl model 1 k = 43, had significant better accuracy for low and high scores than the other two kNN models, however its accuracy for average score was significantly less than the other two models. The two models for kNN with all numeric variables included had similar results even with the slightly different k values. The model with the best overall percentage is the third model.

- **PERCENT PROFIT**

The non-discretized dataset was used to predict percent profit using K Nearest Neighbor. Similarly, all non-numeric variables were removed from the original dataset, as well as gross and profit. The data was split into testing and training data and fed into the KNN model. Multiple models were created, none with significant accuracy. One is shown below.



```
           df_test_label
kNN         average high negative
  average        31   11       21
  high           18   50       26
  negative       25   13       27
```

Figure 81.

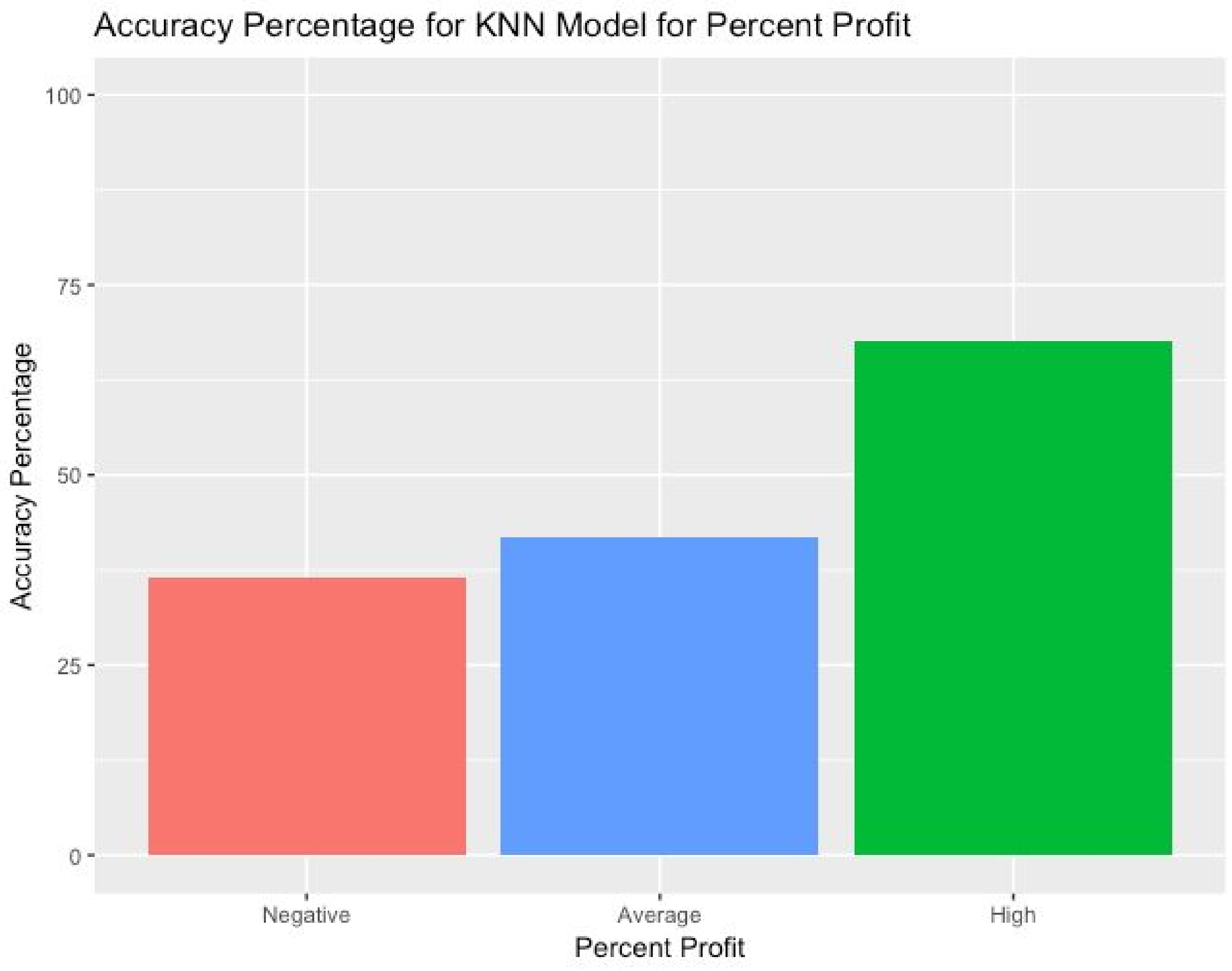


Figure 82.

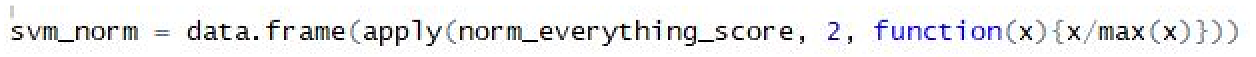
The results varied (but not by much) across all KNN models. Again, these models lack a significant level of accuracy and need additional data in order to be useful.

VII. SVM

SCORE

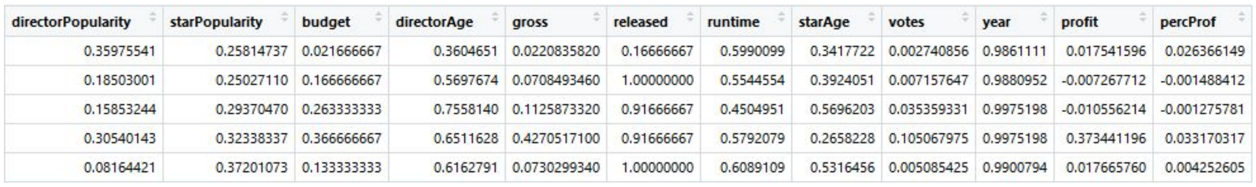
Model 1

The first model utilized a normalized dataset of all numeric variables with the score variable removed. The data frame contains 741 observations of 12 variables.   The data was normalized by dividing every value each column by the maximum value in the column. This resulted with values between 0 and 1.



```
svm_norm = data.frame(apply(norm_everything_score, 2, function(x){x/max(x)}))
```

Figure 82a. Code Snippet to Normalize Dataset

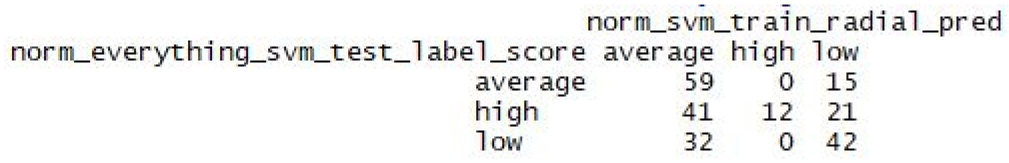


directorPopularity	starPopularity	budget	directorAge	gross	released	runtime	starAge	votes	year	profit	percProf
0.35975541	0.25814737	0.021666667	0.3604651	0.0220835820	0.16666667	0.5990099	0.3417722	0.002740856	0.9861111	0.017541596	0.026366149
0.18503001	0.25027110	0.166666667	0.5697674	0.0708493460	1.00000000	0.5544554	0.3924051	0.007157647	0.9880952	-0.007267712	-0.001488412
0.15853244	0.29370470	0.263333333	0.7558140	0.1125873320	0.91666667	0.4504951	0.5696203	0.035359331	0.9975198	-0.010556214	-0.001275781
0.30540143	0.32338337	0.366666667	0.6511628	0.4270517100	0.91666667	0.5792079	0.2658228	0.105067975	0.9975198	0.373441196	0.033170317
0.08164421	0.37201073	0.133333333	0.6162791	0.0730299340	1.00000000	0.6089109	0.5316456	0.005085425	0.9900794	0.017665760	0.004252605

Figure 83. First 5 Rows of the norm_everything_svm Test DF

The data frame was then split into a training and test dataset. The training dataset contained 519 observations. There were 173 high scores, 173 average scores and 173 low scores. The testing dataset was comprised of 74 high scores, 74 average scores, and 74 low scores for a total of 222 observations. The score label was removed from the testing set.

The first attempt for svm utilized a radial kernel and a cost of .1.



```
                                   norm_svm_train_radial_pred
norm_everything_svm_test_label_score average high low
                             average      59    0  15
                             high         41   12  21
                             low          32    0  42
```

Figure 84. Table of Results for svm_radial Model 1

The model accurately predicted low scores 57%, average scores 80% and high scores 12% of the time. It is interesting to note that the model did not classify any other scores as high, except for the scores that were actually high. The model classified the majority of movies as having an average score.

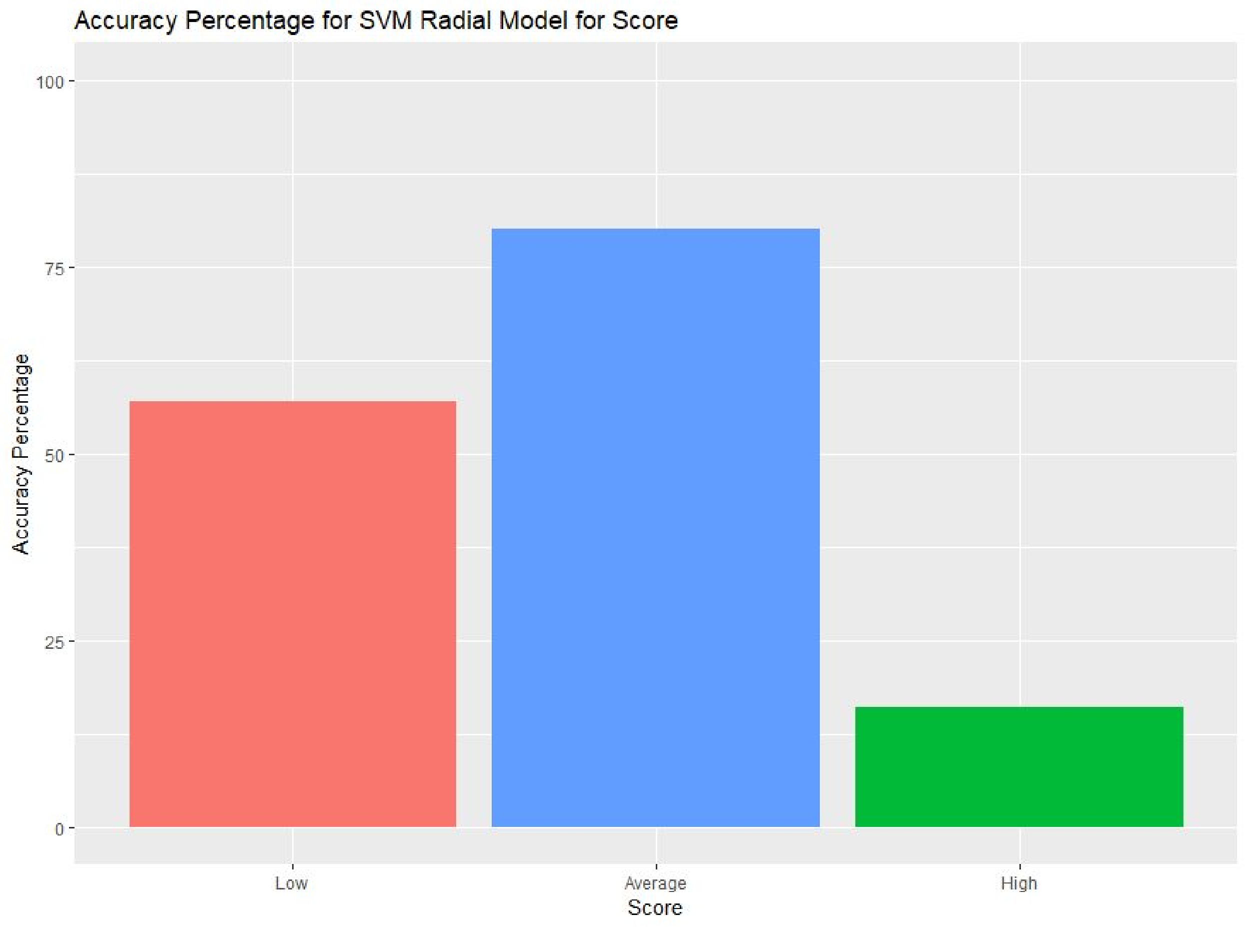
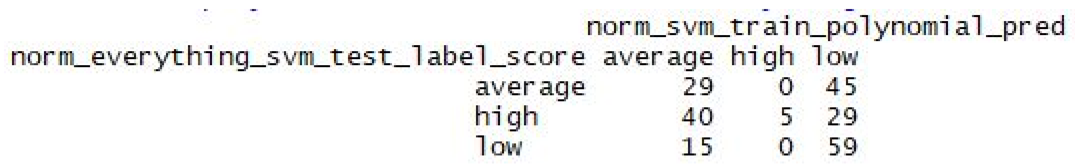


Figure 85. Visualization of Accuracy Percentage for SVM Radial Model for Score

The second attempt at svm utilized the polynomial kernel.



```
                                   norm_svm_train_polynomial_pred
norm_everything_svm_test_label_score average high low
                             average      29    0  45
                             high         40    5  29
                             low          15    0  59
```

Figure 86. Table of Results for SVM Polynomial Linear Kernel

The model had an overall accuracy of 42%. The model successfully classified low scores 80%, average scores 39%, and high scores 7% of the time. The model misclassified average scores as low scores. Only 15 low scores were misclassified as average scores. However high scores were classified as both average and low, with the exception of the 5 scores that were correctly classified.

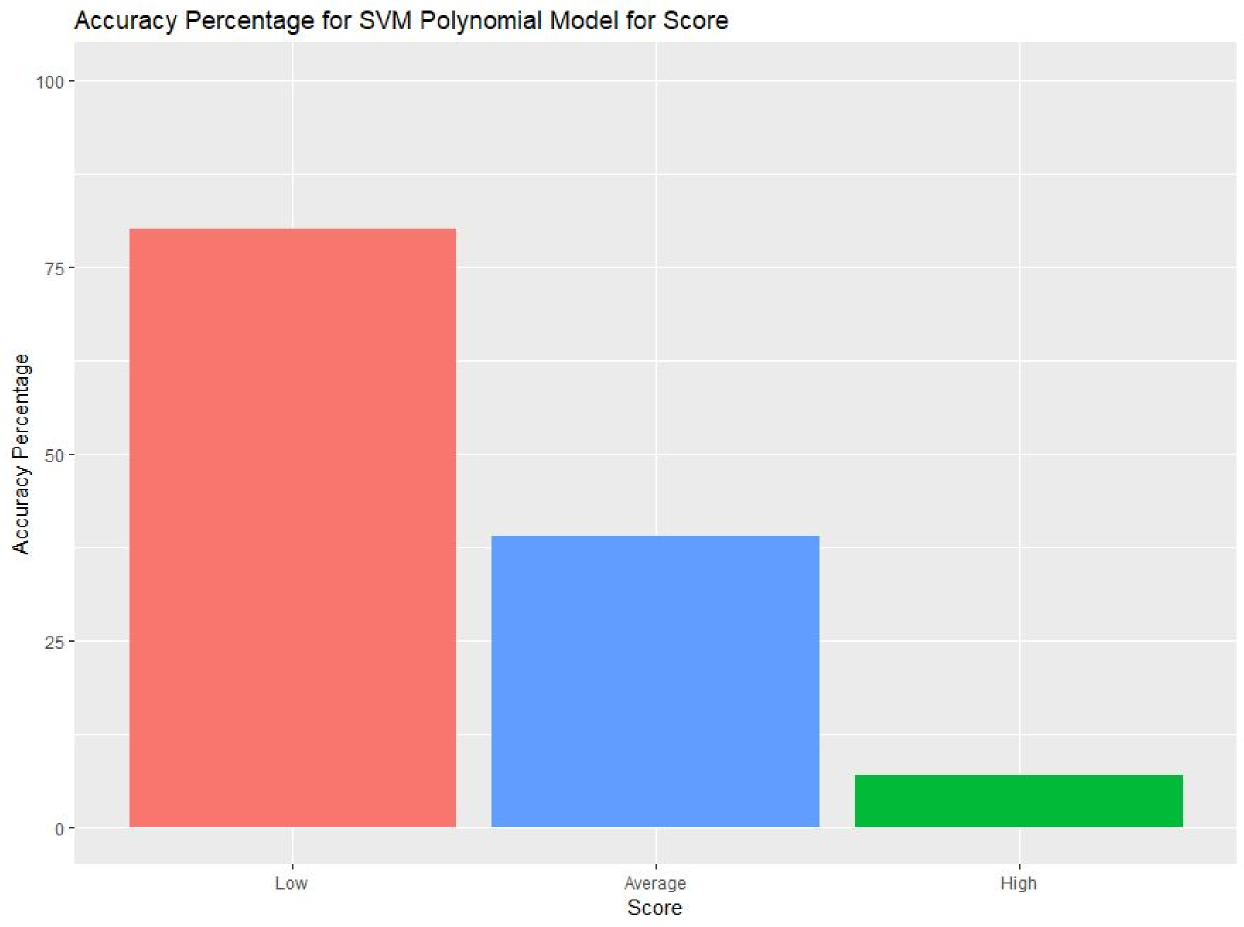
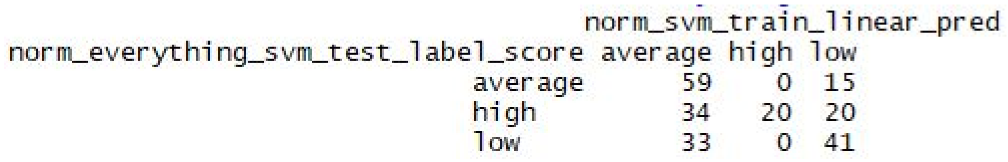


Figure 87. Visualization of Accuracy Percentage for SVM Polynomial Model for Score

The third attempt for svm utilized the linear kernel. This model had an overall accuracy of 54%.



```
                                  norm_svm_train_linear_pred
norm_everything_svm_test_label_score average high low
                             average      59    0  15
                             high         34   20  20
                             low          33    0  41
```

Figure 88. Table of Results for SVM Linear Kernel

The model accurately classified low scores 55%, average scores 80%, and high scores 27% of the time.

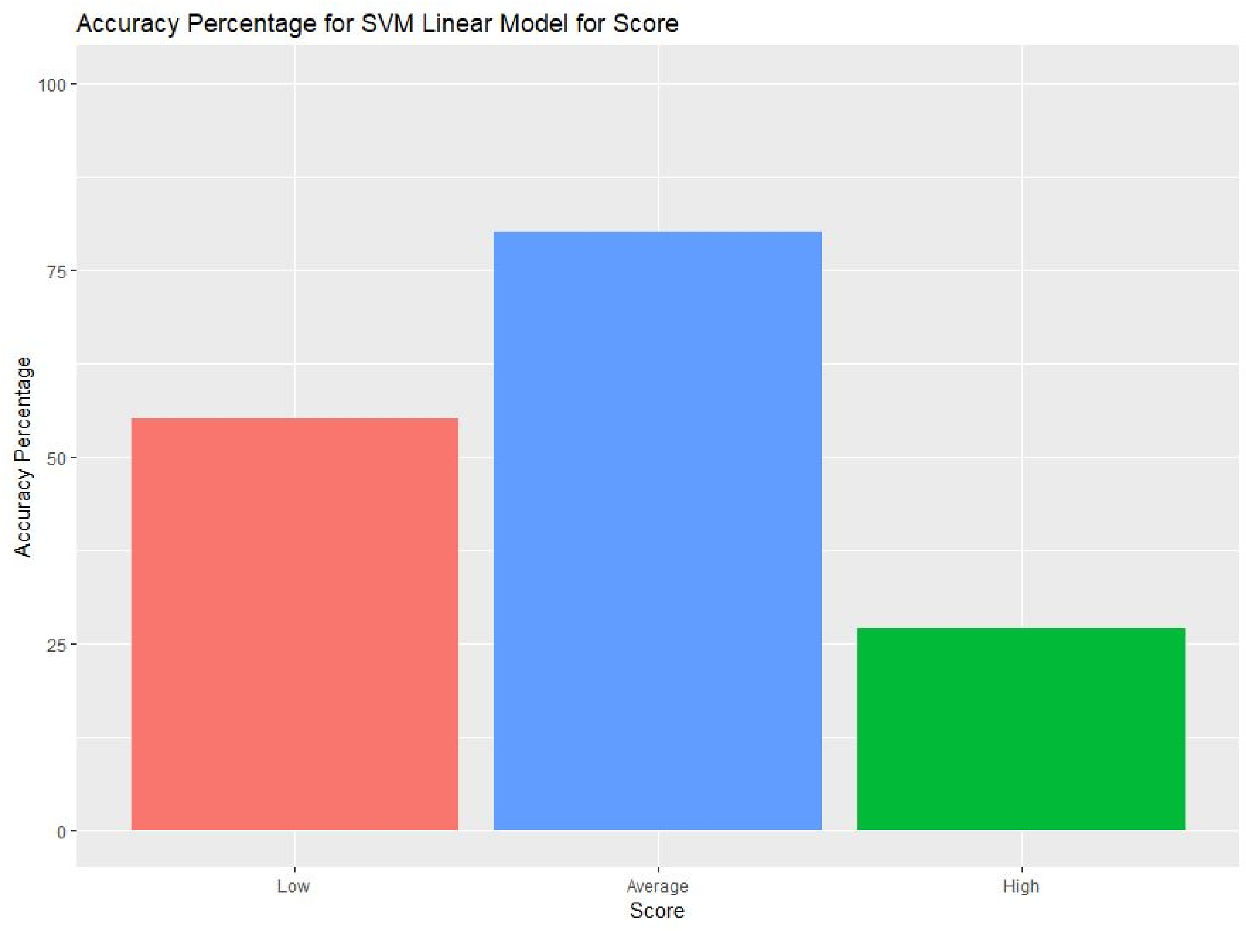


Figure 89. Visualization of Accuracy Percentage for SVM Linear Model for Score

Comparison of All SVM Models for Score

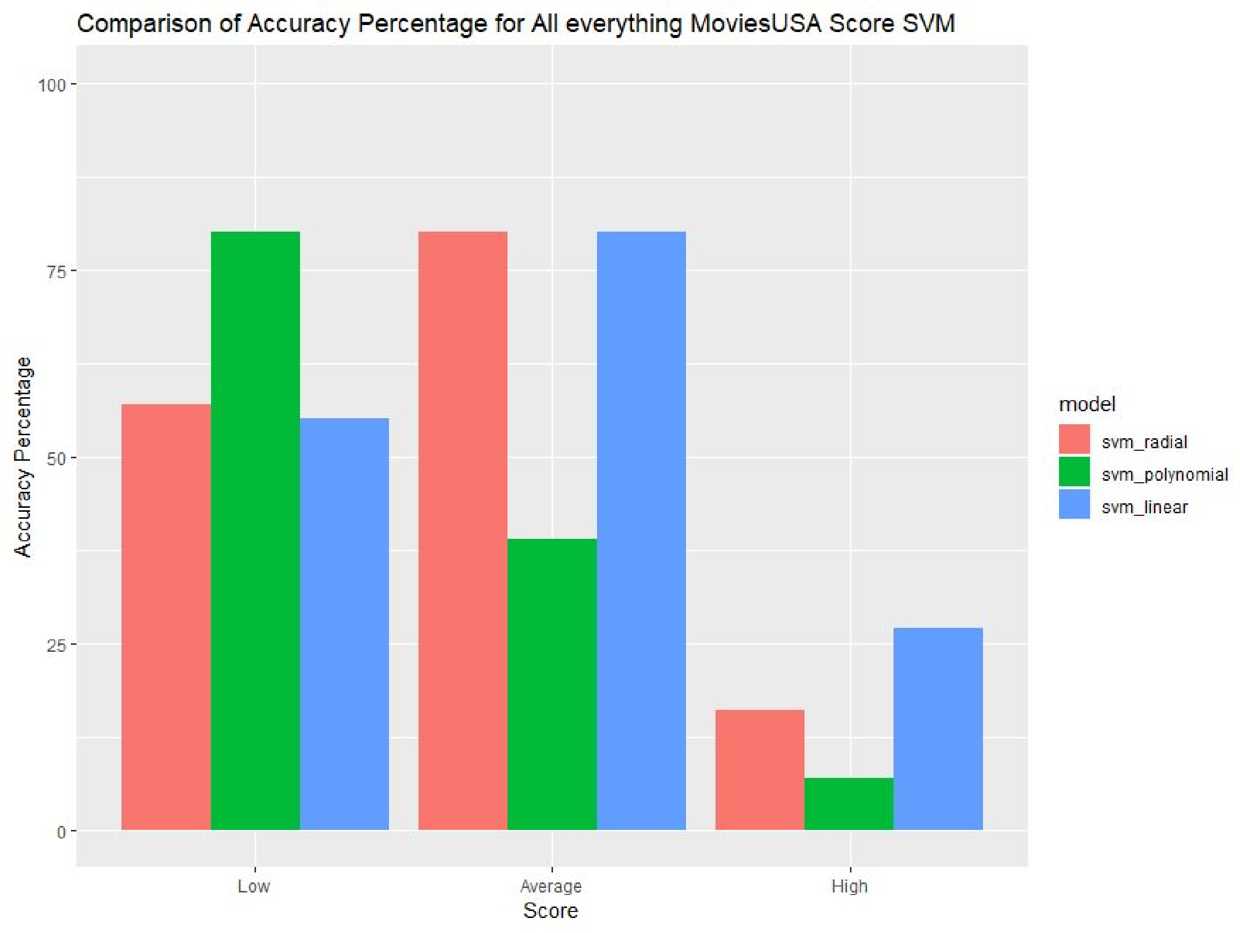


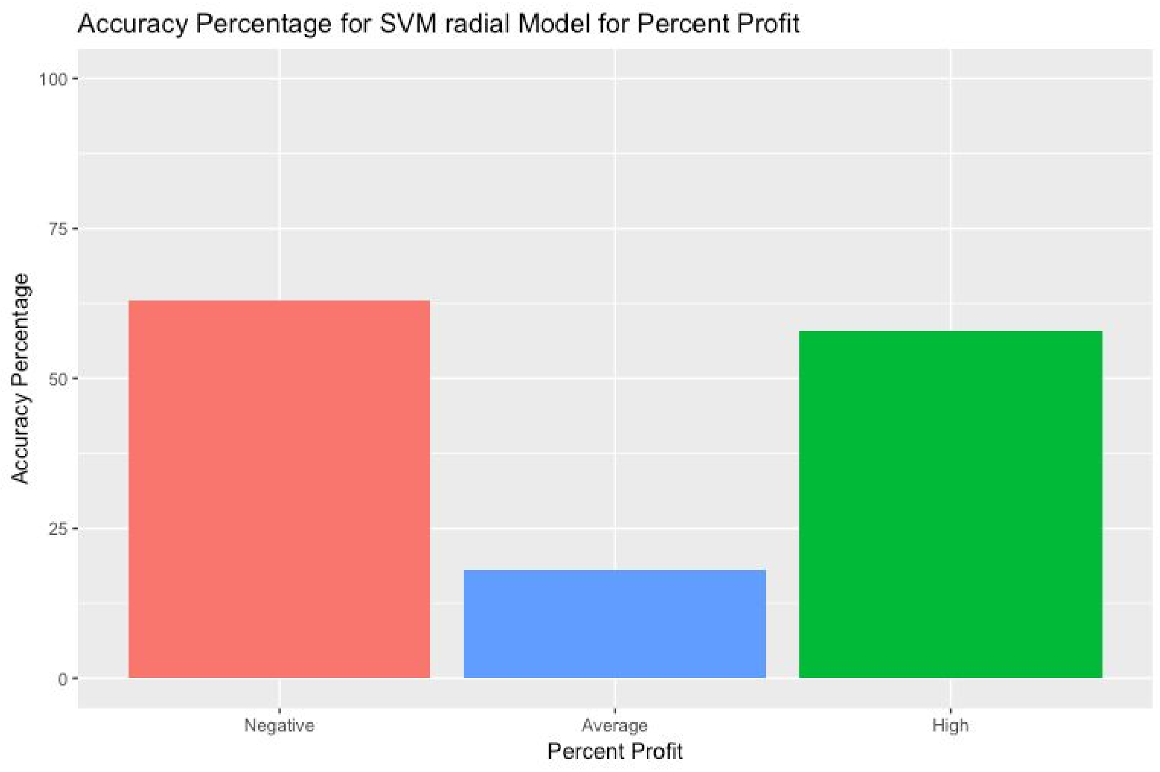
Figure 90. Visualization of Comparison of Accuracy Percentage for All everything MoviesUSA Score SVM

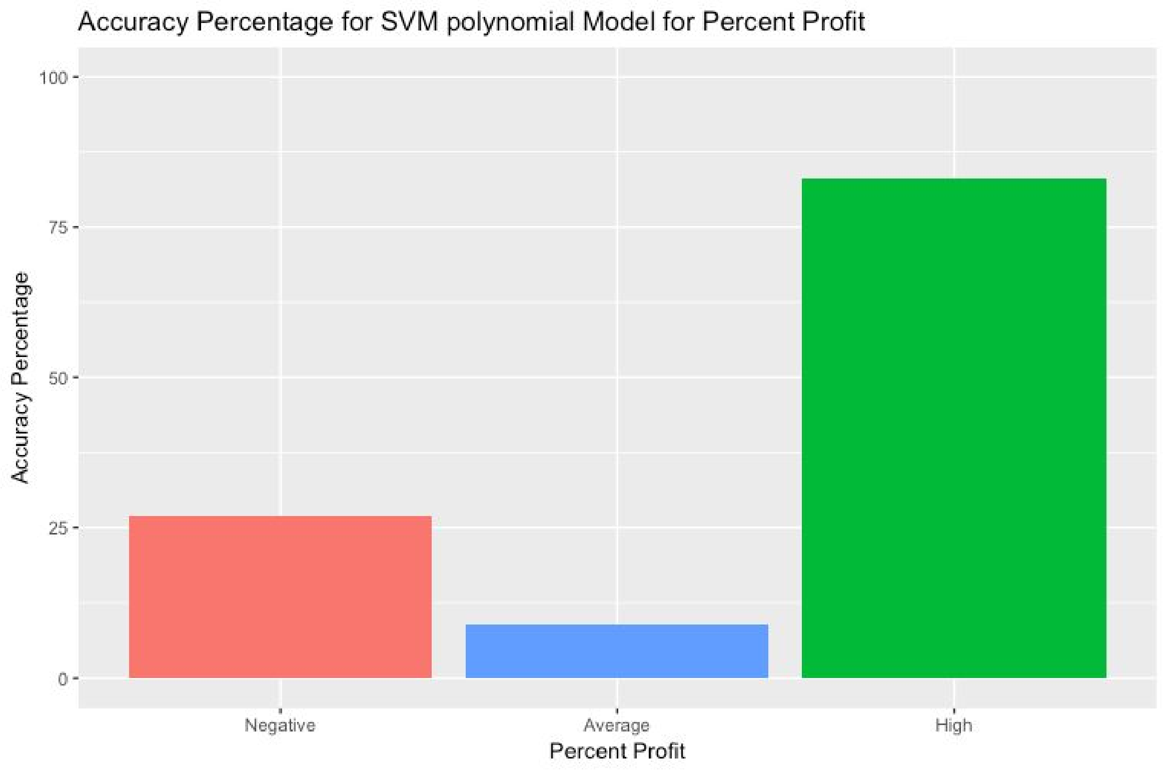
The three svm models with different kernels all had a difficult time correctly classifying movies with a high score. The linear kernel provided the highest accuracy percentage at 54%.

PERCENT PROFIT

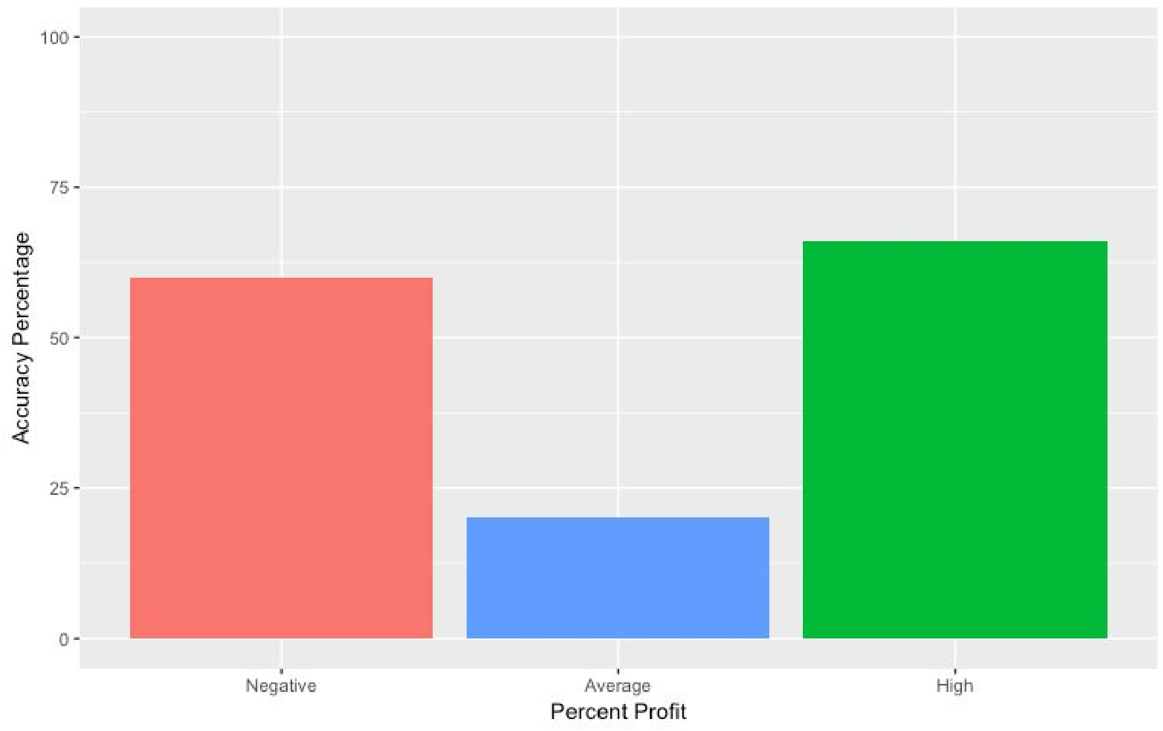
The normalized dataset was used to predict percent profit. Similarly, the variables for title, director, star and released were removed from the original discretized dataset, as well as gross and profit, The data was split into testing and training data and fed into the SVM model.

Figures 91-94.

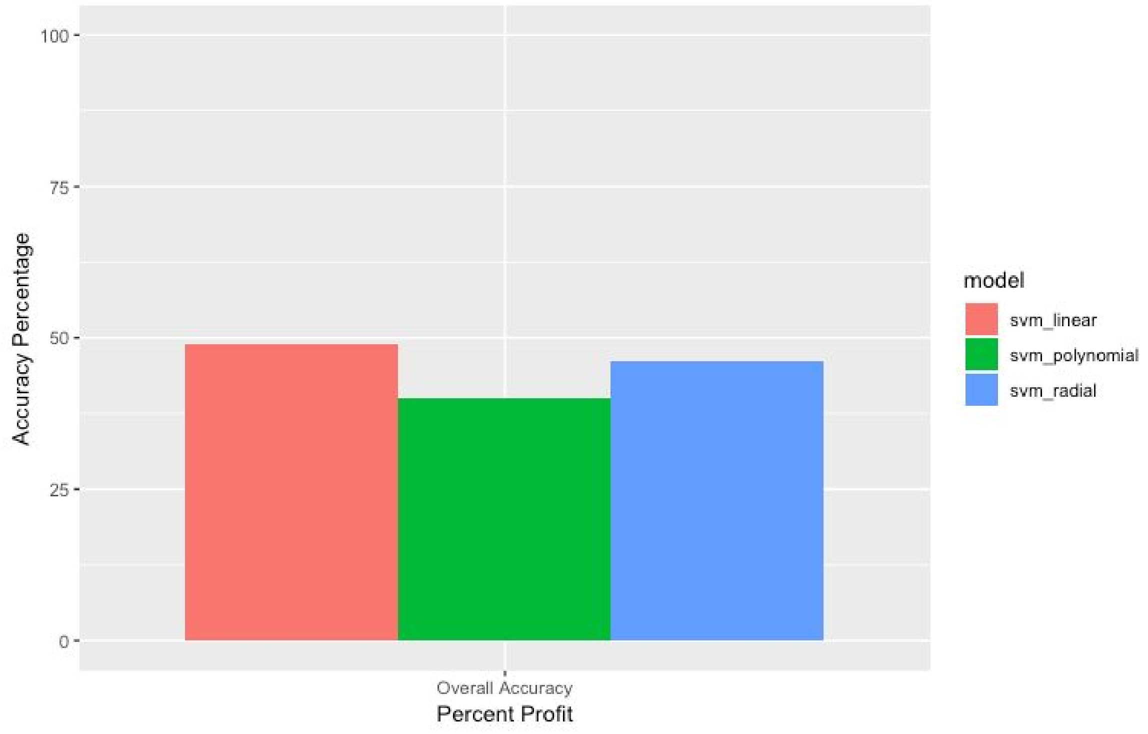




Accuracy Percentage for SVM Linear Model for Percent Profit



Comparison of Overall Accuracy Percentage for All everything MoviesUSA Percent Profit S



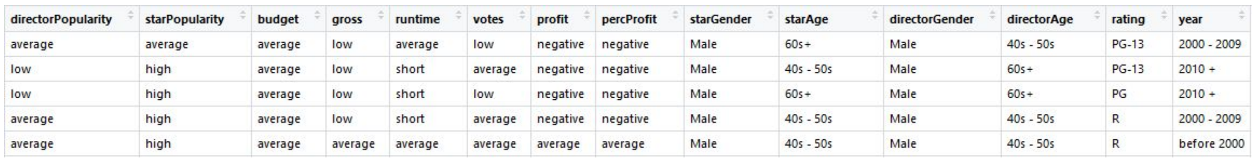
The predictive accuracy across all three SVM models was similar, however, the differences between the "Negative," "Average" and "High" predictions varied greatly between linear, polynomial and radial. Unfortunately, the researchers aren't yet sure how to melt the three different models into one ubermodel, so the fact that SVM polynomial was so adept at predicting high profits is lost to the world until future models (likely with different and additional data) can be run.

VIII. RANDOM FOREST

SCORE

Model 1

The first model for random forest utilized a fully discretized data frame, with 741 observations of 15 variables. The variables for title, director, star, released and genre were removed from the original discretized dataset.

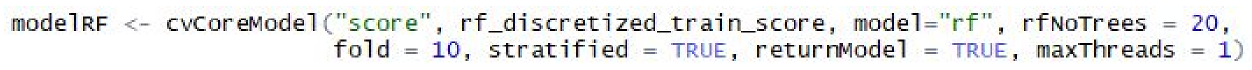


directorPopularity	starPopularity	budget	gross	runtime	votes	profit	percProfit	starGender	starAge	directorGender	directorAge	rating	year
average	average	average	low	average	low	negative	negative	Male	60s+	Male	40s - 50s	PG-13	2000 - 2009
low	high	average	low	short	average	negative	negative	Male	40s - 50s	Male	60s+	PG-13	2010 +
low	high	average	low	short	low	negative	negative	Male	60s+	Male	60s+	PG	2010 +
average	high	average	low	short	average	negative	negative	Male	40s - 50s	Male	40s - 50s	R	2000 - 2009
average	high	average	average	average	average	average	average	Male	40s - 50s	Male	40s - 50s	R	before 2000

Figure 95. First 5 Rows of the rf_discretized_train_score df

The data frame was then split into a training and test dataset. The training dataset contained 519 observations. There were 173 high scores, 173 average scores and 173 low scores. The testing dataset was comprised on 74 high scores, 74 average scores, and 74 low scores for a total of 222 observations. The score label was removed from the testing set.

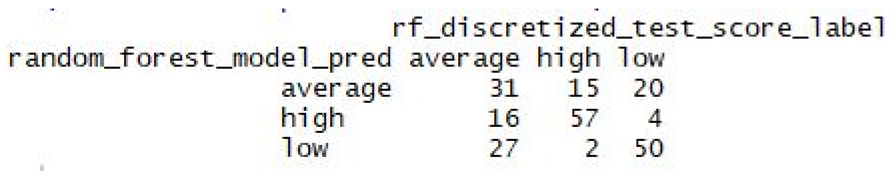
The first model for random forest utilized the CORELearn package and used the cvLearn function to predict 20 trees with a 10-fold cross validation. The cvLearn is more appropriate for a large data set and can specify just the target variable.



```
modelRF <- cvCoreModel("score", rf_discretized_train_score, model="rf", rfNoTrees = 20,
                       fold = 10, stratified = TRUE, returnModel = TRUE, maxThreads = 1)
```

Figure 96. Code Snippet for cvCoreModel Random Forest

The code in figure 96 states the target variable is score, from the rf_discretized_train_score data frame, the model = “rf”, random forest, the number of trees to create is 20 with a cross fold validation of 10 and to return the best model.



```
                        rf_discretized_test_score_label
random_forest_model_pred average high low
                 average      31   15  20
                 high         16   57   4
                 low          27    2  50
```

Figure 97. Table of Results Random Forest CV Model

The model had an overall accuracy of 61%. It correctly identified low scores 70%, average scores 30%, and high scores 77% of the time.

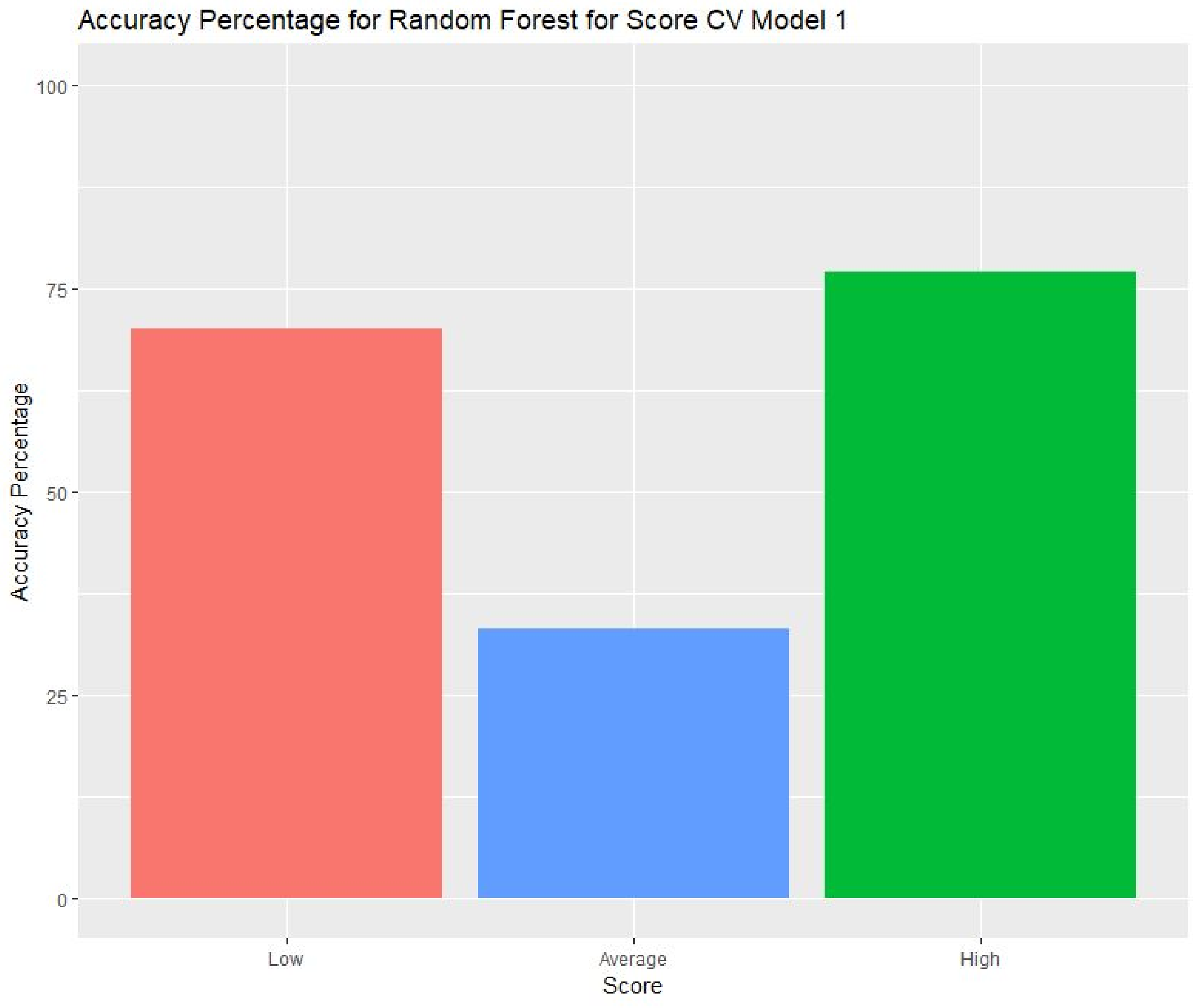


Figure 98. Visualization of Accuracy Percentage for Random Forest for Score CV Model 1

cvLearn also has the ability to evaluate the attributes.

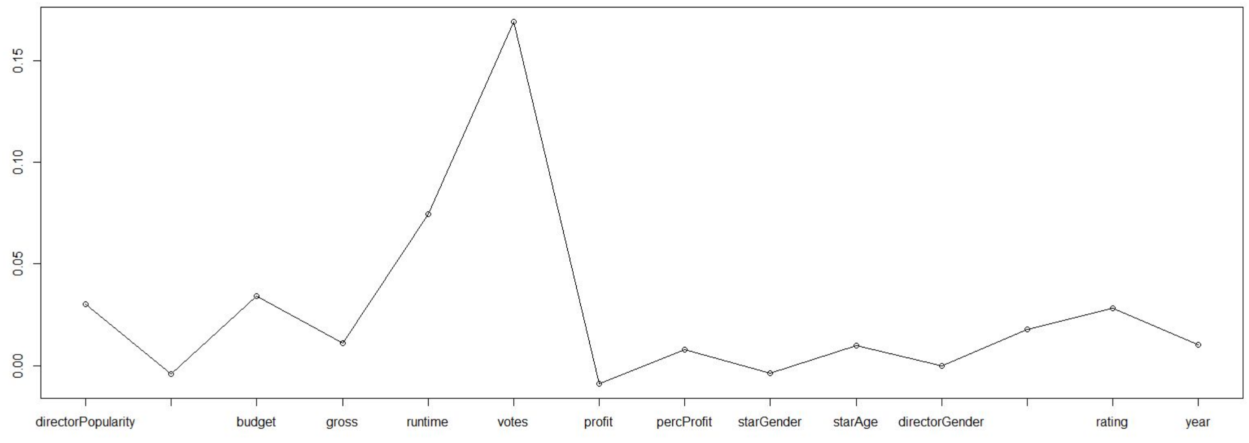
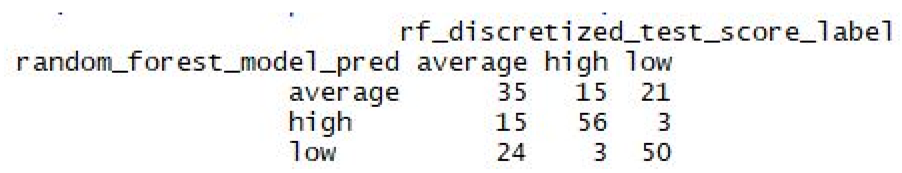


Figure 99. Attribute Evaluation of Attributes for Random Forest

The attributes that were shown to have the most importance in the random forest model are: votes, runtime, budget, and directorPopularity. These will be the only four attributes included in model 2.

The second attempt at random forest utilized the randomForest function.



```
                        rf_discretized_test_score_label
random_forest_model_pred average high low
                 average      35   15  21
                 high         15   56   3
                 low          24    3  50
```

FIgure 100. Table of Results for Random Forest Model

The model had an overall accuracy of 64%. It correctly identified low scores 68%, average scores 47%, and high scores 76% of the time.

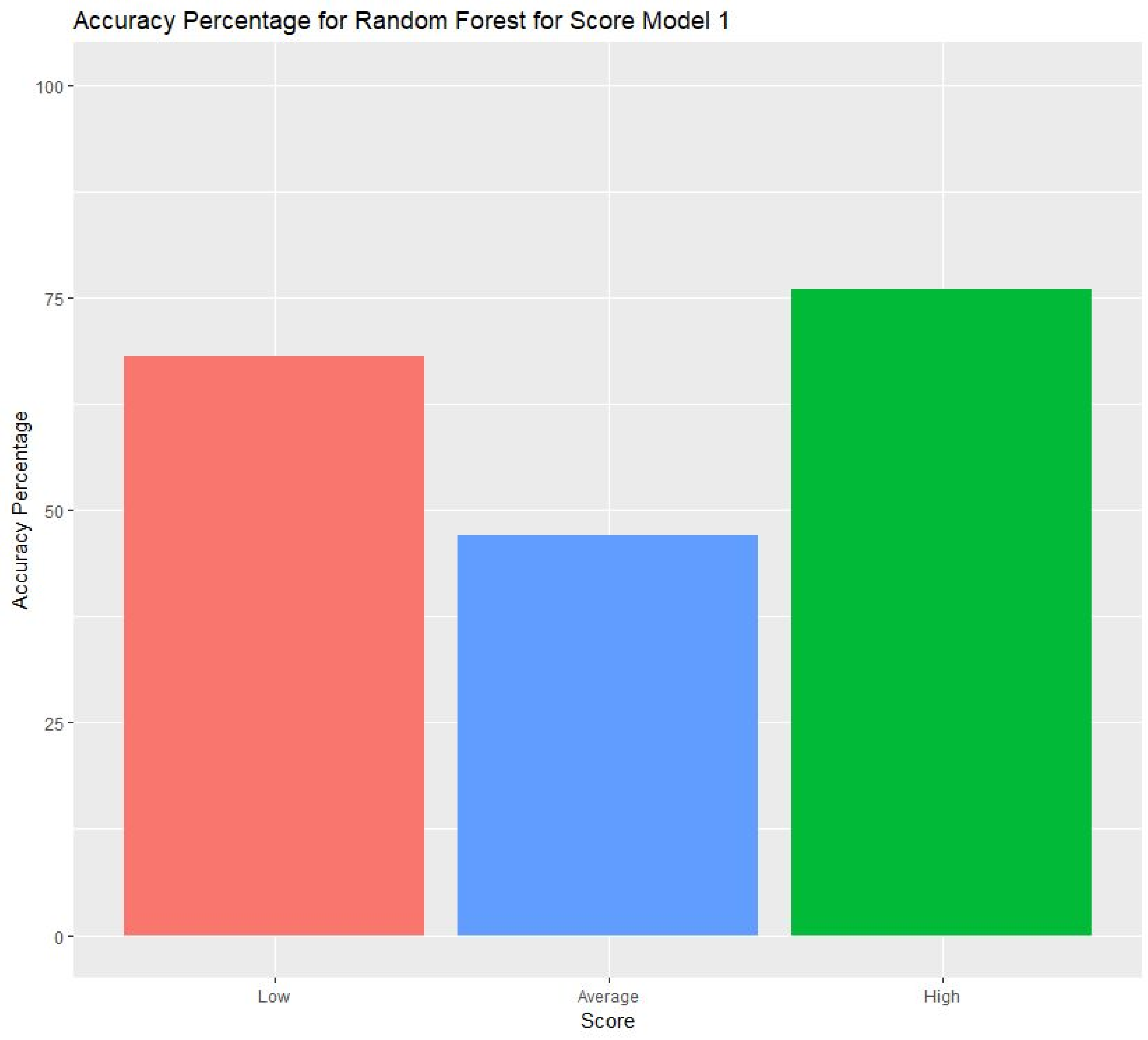
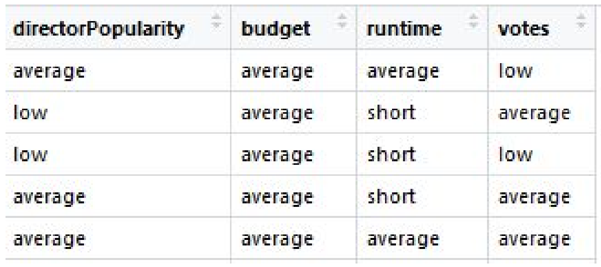


Figure 101. Visualization of Accuracy Percentage for Random Forest for Score Model 1

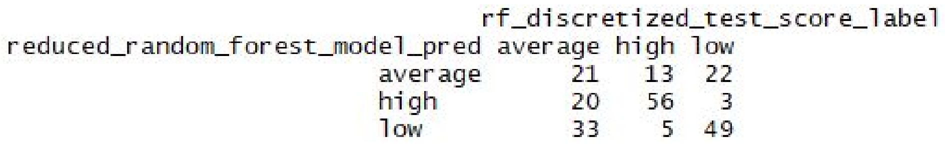
Model 2

The second model reduced the dataset from model 1 to include only the top attributes determined from the attribute evaluation in cvLearn. The model includes only votes, runtime, budget, and directorPopularity.



directorPopularity	budget	runtime	votes
average	average	average	low
low	average	short	average
low	average	short	low
average	average	short	average
average	average	average	average

Figure 102. First 5 Rows of the Reduced Data Frame for Random Forest



```
                                  rf_discretized_test_score_label
reduced_random_forest_model_pred average high low
                         average      21   13  22
                         high         20   56   3
                         low          33    5  49
```

Figure 103. Table of Results for the Reduced Random Forest Model 2

The model had an overall accuracy of 56%. It correctly identified low scores 66%, average scores 20%, and high scores 76% of the time.

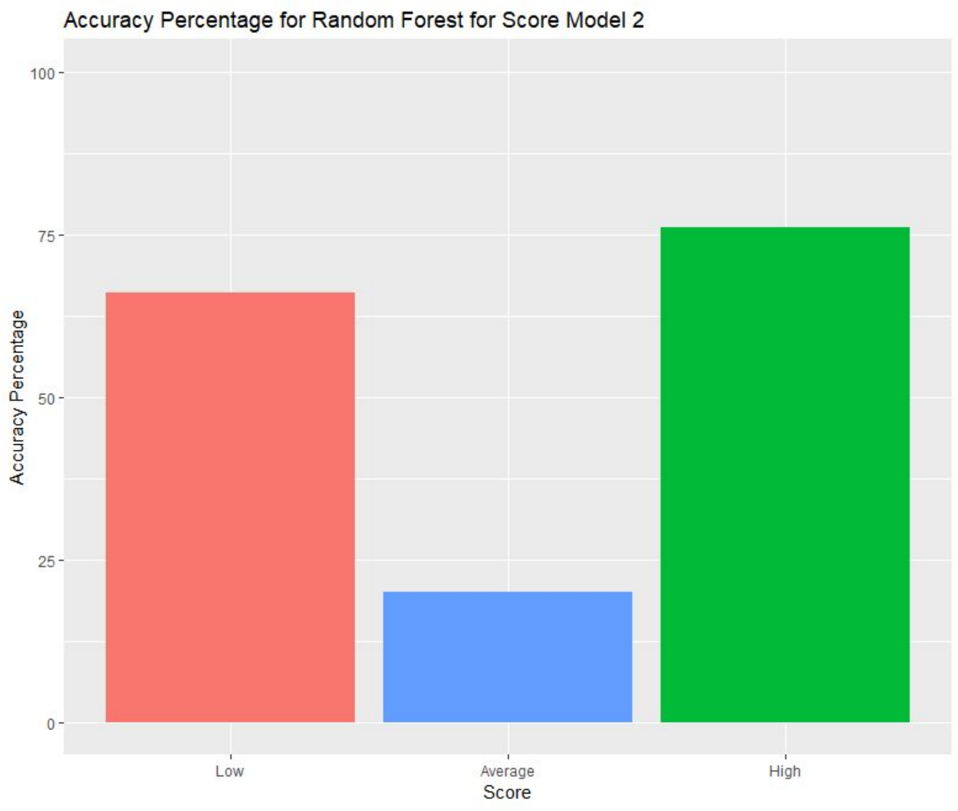


Figure 104. Visualization of Accuracy Percentage for Random Forest for Score Model 2 - Reduced Dimensions

Comparison of Random Forest Models for Score

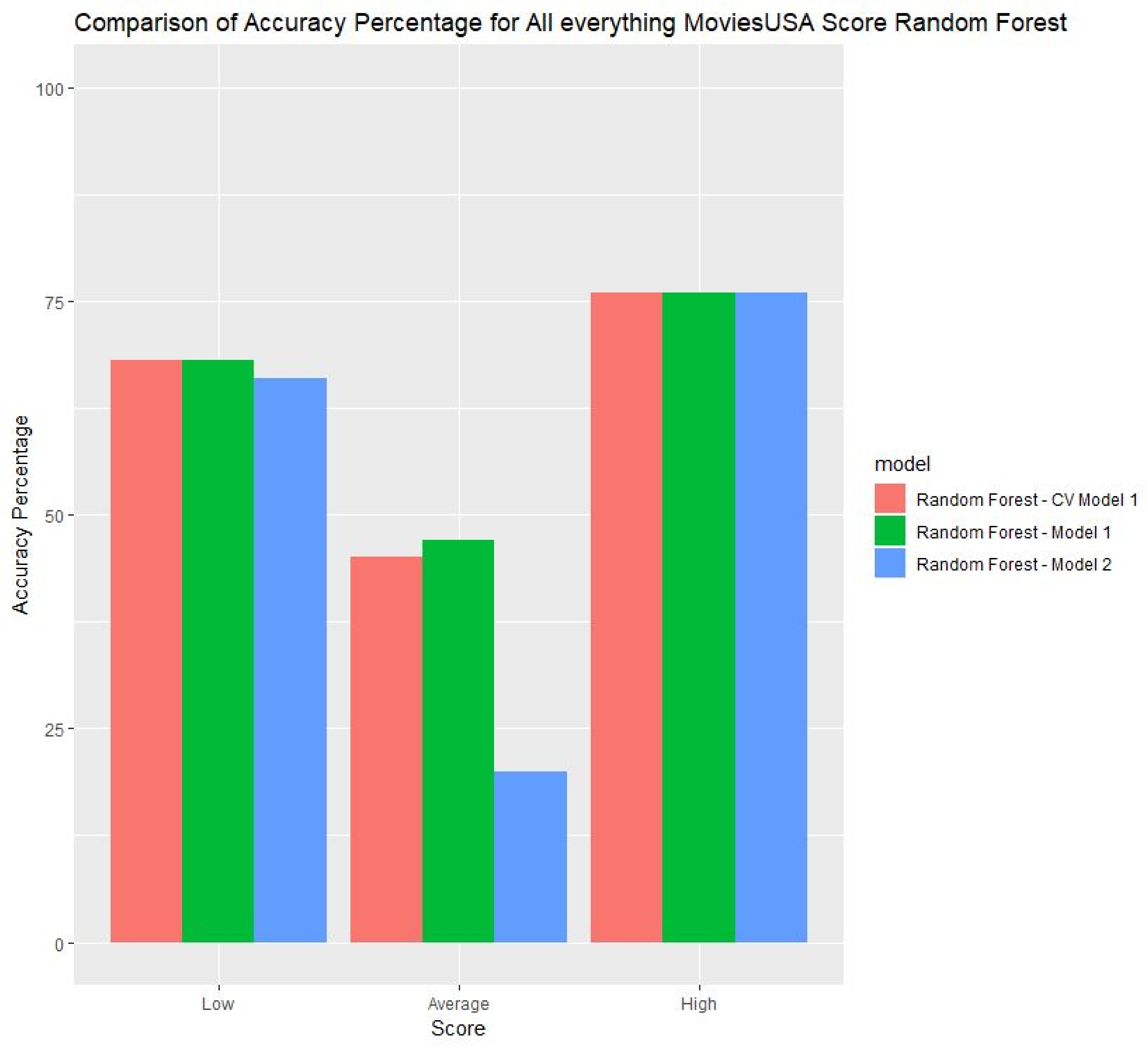


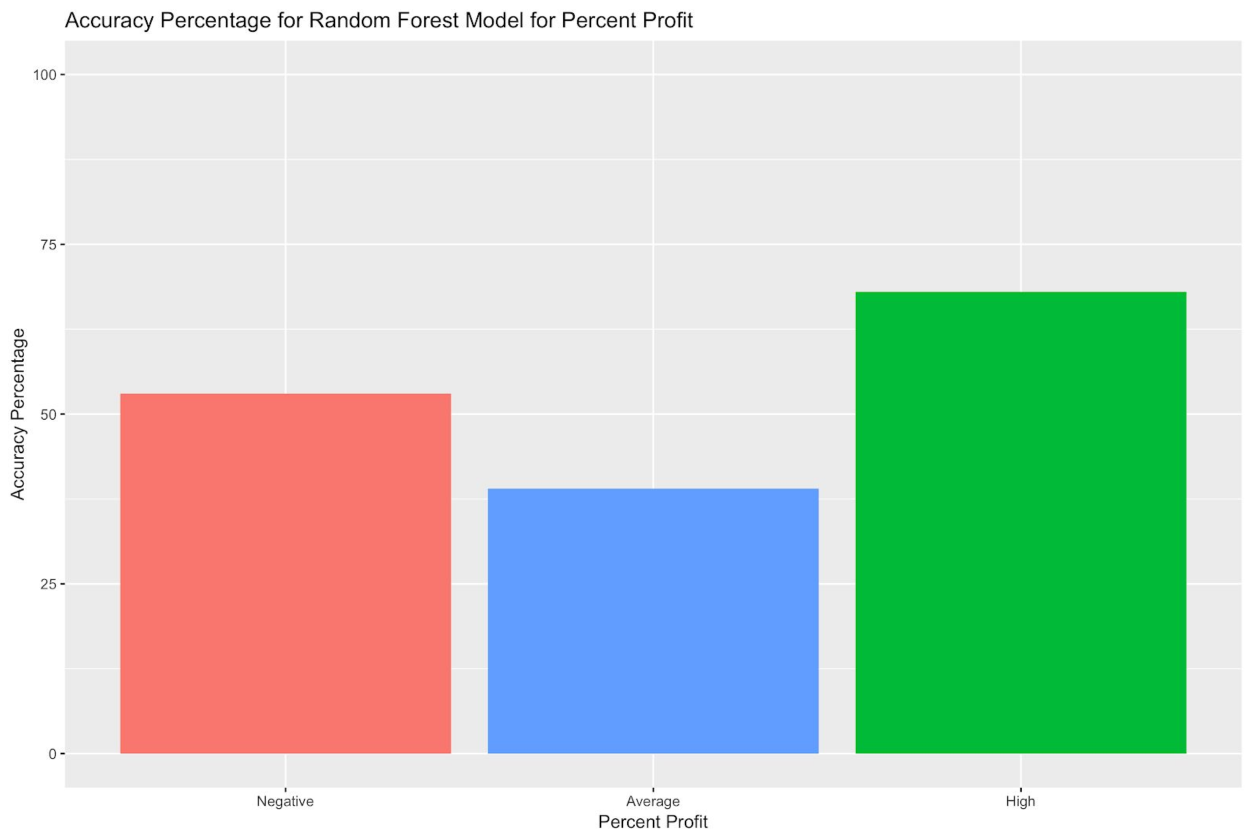
Figure 105. Comparison of Accuracy Percentage for ALl Everything MoviesUSA Score Random Forest

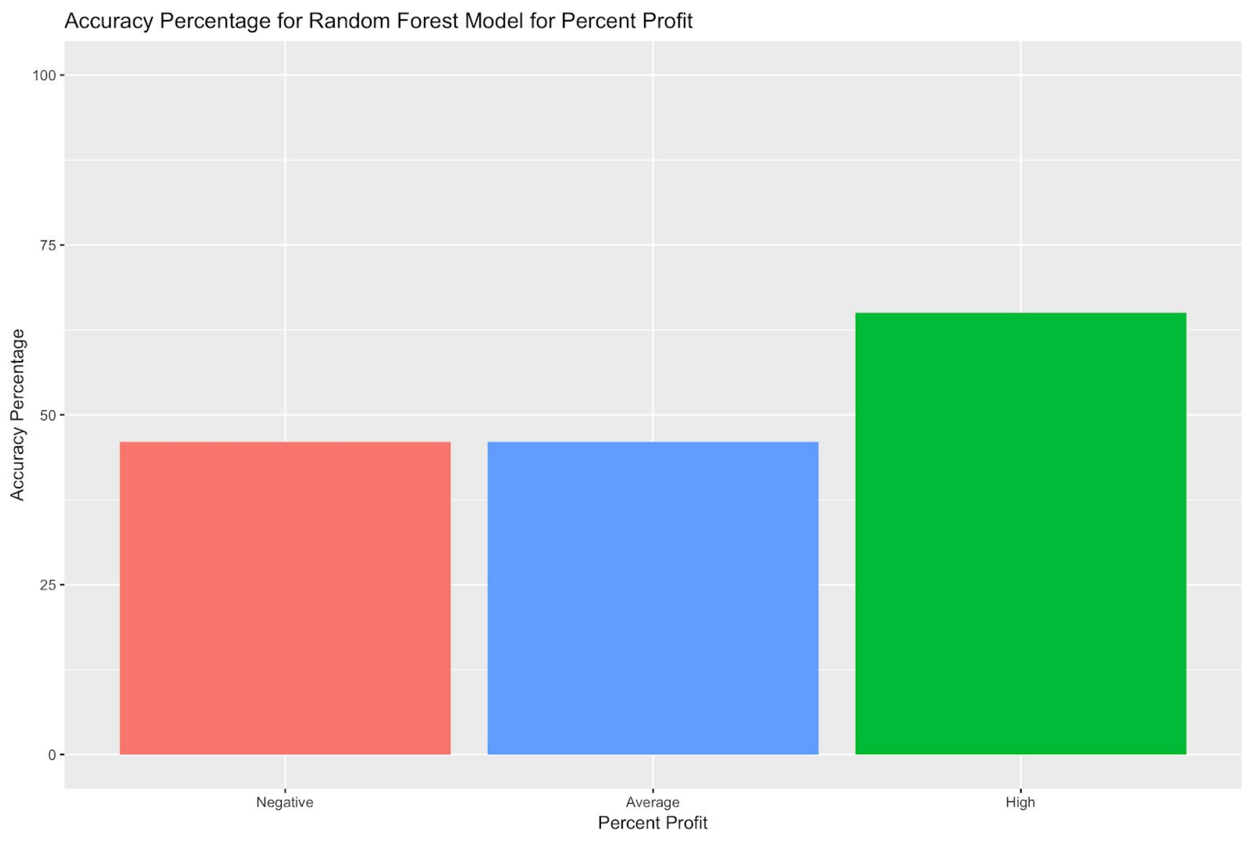
All models appeared to have the same accuracy for classifying high scores and similar accuracy when classifying low scores. The difference in the models is apparent when classifying average scores. The random forest model with reduced dimensions was not successful in this classification. The CV model 1 and random forest model 1 were both more successful, but the random forest model 1 was the most successful. Ultimately the random forest model 1 proved to have the highest overall accuracy at 64%

PERCENT PROFIT

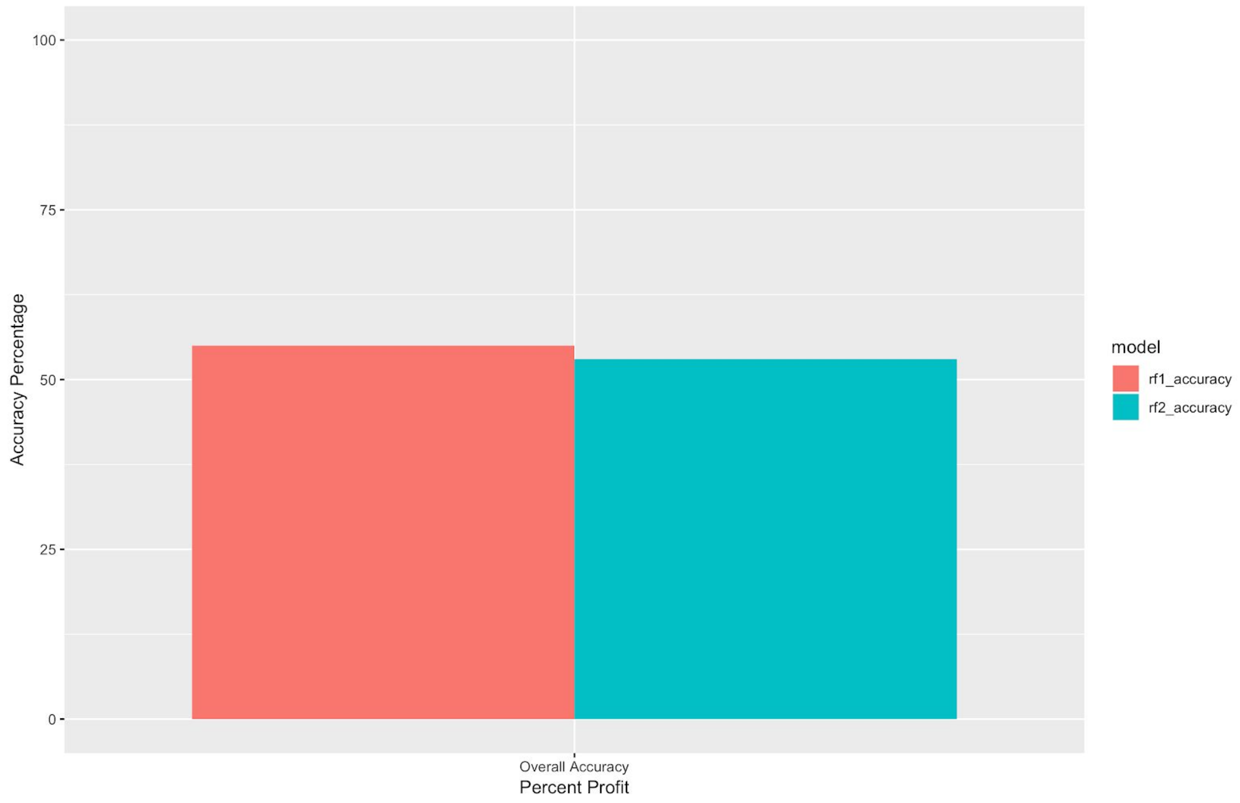
Both models for random forest utilized the same fully discretized data frame, with 741 observations of 15 variables. Just like with score, the variables for title, director, star, released and genre were removed from the original discretized dataset.

Figures 106-109.

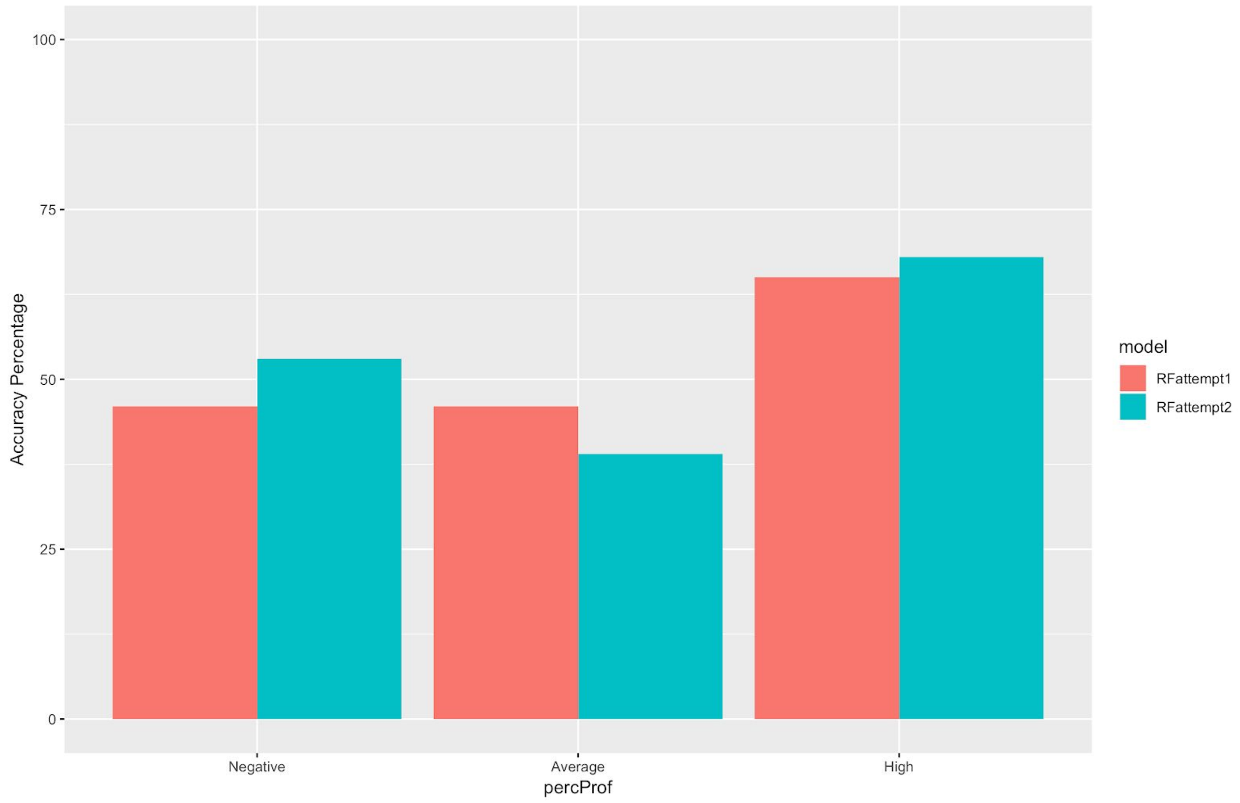



Accuracy Percentage for Random Forest Model for Percent Profit
Accuracy Percentage
100
75
50
25
0
Negative
Average
High
Percent Profit

Comparison of Overall Accuracy Percentage for All everything MoviesUSA Percent Profit RF



Comparison of Accuracy Percentage for All everything MoviesUSA Percent Profit RF



Once again, despite being pushed and prodded into every possible model, the dataset did not yield reveal its secrets to the researchers. So far, the models have proved educational but inconclusive. Despite being downtrodden with continued meidiogracy from their beloved models, the researchers continue to enthusiastically search for clues to the hidden treasures within the Internet Movie Database.

IX. TEXT-MINING

"Heigh-ho, heigh-ho, it's off to work we go"

-Seven anonymous miners (not minors)

The data was run through the baseline text-mining r-script.

```r
library(tm)
library(wordcloud)
file <- "your-file-here"
fileData <- readLines(file)
words.vec <- VectorSource(fileData)
words.corpus <- Corpus(words.vec)
words.corpus <- tm_map(words.corpus, content_transformer(tolower))
words.corpus <- tm_map(words.corpus, removePunctuation)
words.corpus <- tm_map(words.corpus, removeNumbers)
words.corpus <- tm_map(words.corpus, removeWords,
stopwords("english"))
tdm <- TermDocumentMatrix(words.corpus)
m <- as.matrix(tdm)
wordCounts <- rowSums(m)
wordCounts <- sort(wordCounts, decreasing=TRUE)
wordcloud(names(wordCounts), wordCounts)

wordcloud(names(wordCounts), wordCounts, min.freq=2, max.words=50,
rot.per=0.35, colors=brewer.pal(8, "Dark2"))
wordToAssociate <- "your-word-here"
findAssocs(tdm, wordToAssociate, 0.2)
```

Figure 110.

Figure 110 was the result.

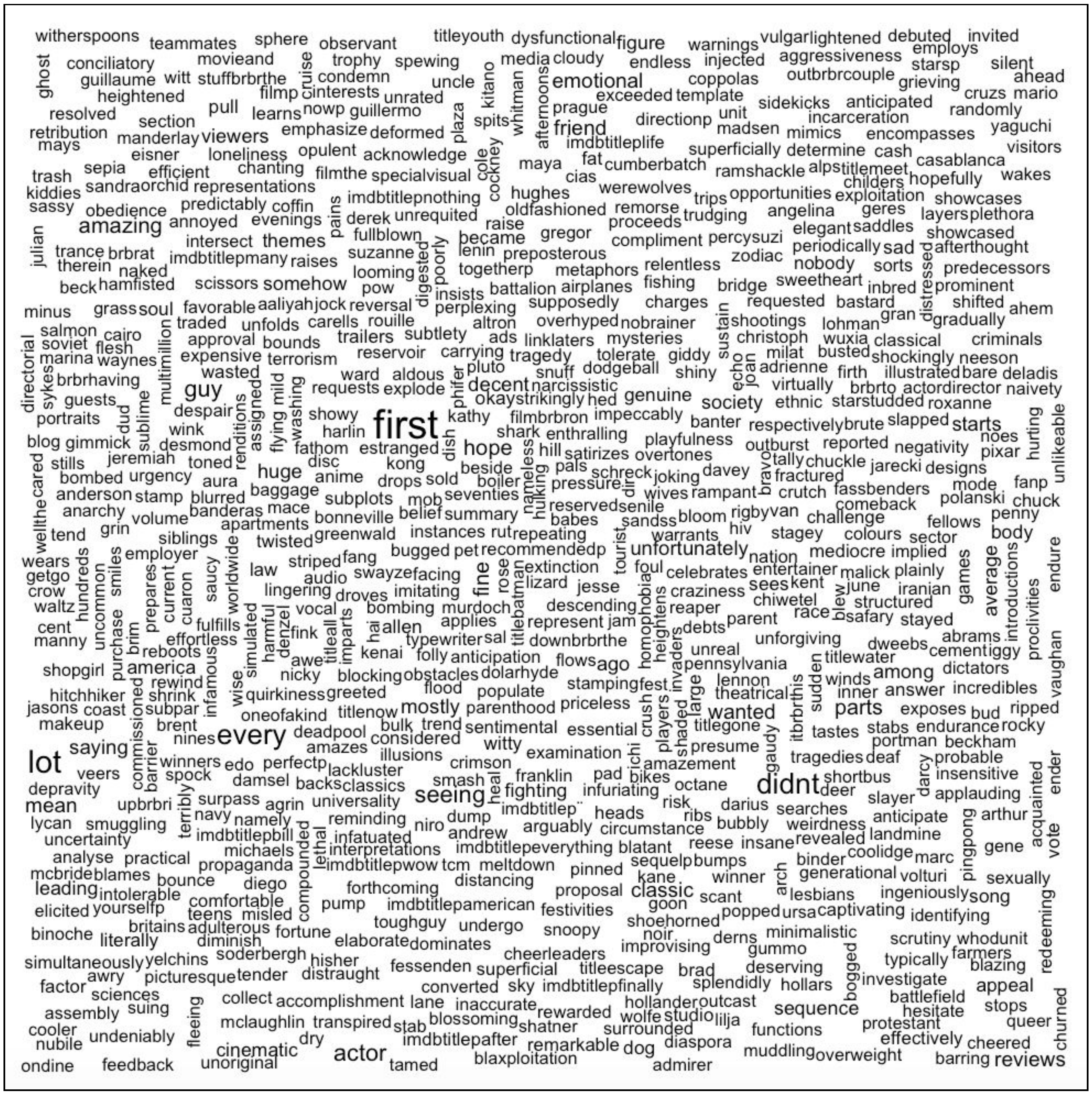

first
lot
every
didnt
actor
seeing
guy
amazing
emotional
friend
decent
hope
unfortunately
parts
wanted
classic
reviews
society
starts
huge
mostly
cinematic
sequence
appeal
body
average
fine
somehow
themes
viewers
witherspoons
teammates
sphere
observant
titleyouth
dysfunctional
figure
warnings
vulgar
lightened
debuted
invited
employs

Figure 111.

After refining by removing punctuation and stopwords, Figure 112. was the result.

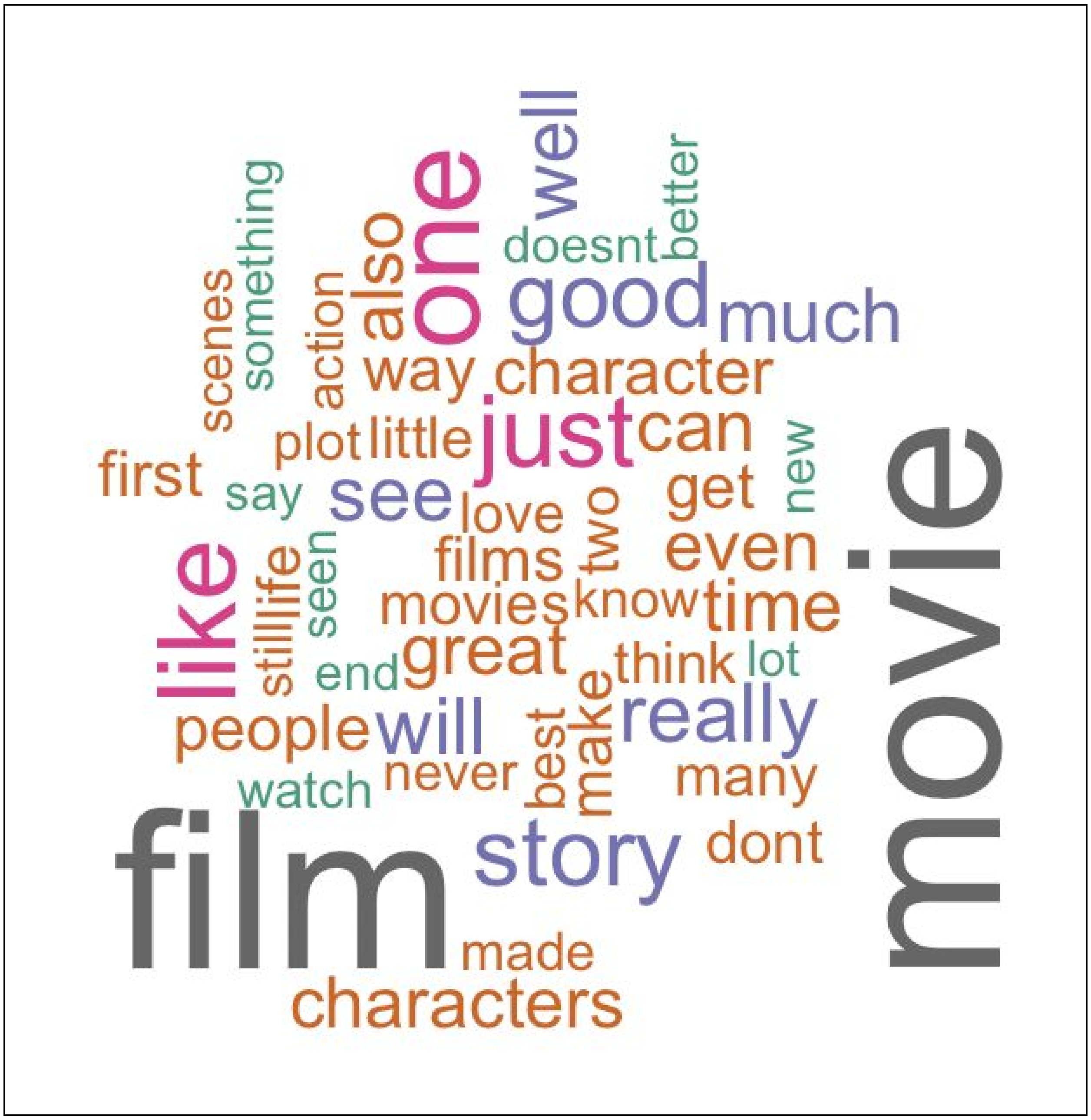


Figure 112.

The researchers had grand plans regarding text mining. Their ultimate goal with this dataset was to determine whether early reviews were a potential indicator of box-office flops. Instead of running any predictive algorithms on this dataset, the researchers chose to rely solely on text mining. Sentiment analysis was applied to the mined dataset, and the results ran counter to the researchers' original hypothesis. In technical terms, 6% of the words in the reviews corresponded to the "negative.txt" document used in the sentiment analysis, while 9% of the words corresponded to the "positive.txt" document. In short, when analyzed as a whole, the reviews were 6% negative and 9% positive.

X. COMPARISON OF ALL MODELS FOR SCORE

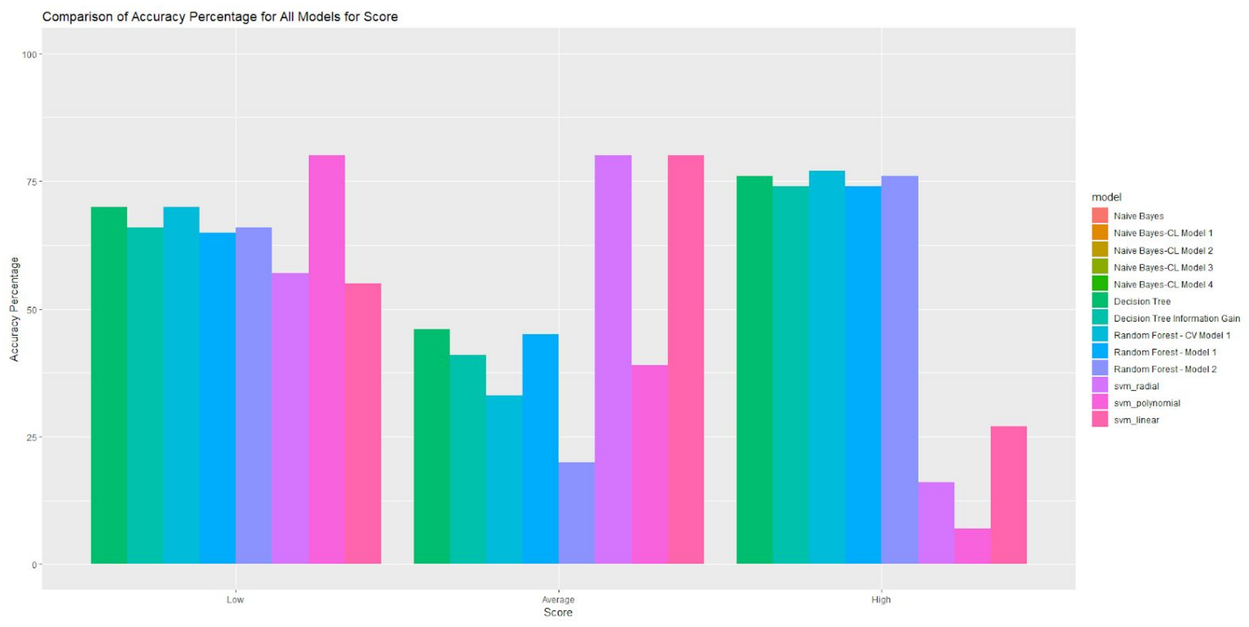


Figure 113. Comparison of All Models for Score

Random Forest and Decision Trees were the best predictors for score. Ultimately, the majority of the models had a difficult time distinguishing average scores. It is important to reexamine how the scores were discretized and take into account that the average score had a very small range. For future analysis, the scores might be discretized differently, and ideally more data would be scraped about the movies.

Conclusion

Initially, the researchers set out to answer three major questions. These questions were intentionally vague and set lofty goals to facilitate a more thorough exploration of the methods employed, as well as lay the groundwork for future studies on this topic. First, can this project recommend movies based on watch history? Without relevant viewing data, the answer remains inconclusive. As an alternative, however, the project can recommend movies based on other criteria, including specific actors, directors, genres, ratings, runtime, and more. For instance, a viewer interested in action movies will likely enjoy director Michael Bay's work, while a viewer looking for a PG-13 comedy without regard to user ratings will likely enjoy movies featuring actor Adam Sandler.

Second, can this project help independent film producers optimize their limited budgets to meet their desired goals? This is where the researchers achieved the most success. The researchers tailored both outcome variables (score and percent profit) to make recommendations on two objectives indie filmmakers would most likely optimize for: achieving high ratings, and saving money. While the project's initial models are not as accurate as the researchers hoped, they create a foundation that can be used to provide more concrete recommendations in the future. Association rule mining, on the other hand, provided much more confident recommendations. For instance, filmmakers looking for a high user rating with a low budget should make an R-rated crime drama, while those looking for more revenue should work in the action genre. (If, for some reason, a filmmaker wanted to produce a high-cost, low-scoring movie, Eddie Murphy is their ideal star.)

Finally, can this project predict a critical failure based on textual analysis of early reviews? Based on time and data limitations, the researchers quickly modified this goal and reached their conclusions through the aforementioned models, rather than text mining. The new question was: can this project accurately predict the IMDB user rating of a given movie based on a series of data points? After combining the gross revenue of a given movie with the average score of the director and lead actor's past movies using Naive Bayes, the project was able to predict with 81% accuracy whether that movie achieved a user rating of 6 or higher.

Movies represent both a significant challenge and a significant opportunity when it comes to data analysis. The movie industry is one of the most significant industries worldwide, both in terms of cultural impact and total revenue generated. In the United States alone, movies made a combined $43 billion in 2018. Yet the growth of this industry is slowing, with studios relying on fewer “tentpole” films to make more of their profits. This strategy carries considerable risk, as a “flop” could have even more significant consequences than in the past. With new and better data analysis tools, filmmakers could work more effectively to produce movies that generate maximum financial and critical success for a minimum of risk, all while making a lasting and positive contribution to our culture.