Predicting IMDB Movie Ratings Using Classification Techniques

Group ID - G5

Introduction

The IMDB database contains a list of movies released up to date. Based on votes from users, each movie is rated on a scale of 10 for its overall performance and likeability. The success of a movie is based on various factors, such as the cast of the movie, director, budget, length of the movie, year of release, genre etc.

In this project we hope to identify the impact of various factors on the rating of a movie and make a prediction on the possible rating of a new movie when released. The ideal model will be able to take any of the above factors or combination of factors as input and estimate the likely rating of a movie.

For the purpose of analysis, the rating of the movies has been classified into three categories. Movies that fall below 6.5 in the IMDB rating have been classified as 3, movies in the range of 6.5 to 7.3 have been classified as 2 and movies above 7.3 in the rating are classified as 1. This classification was performed keeping in mind the quantiles of the ratings. The three categories are, movies below the median, between median and third quantile and above third quantile respectively.

As this is a classification problem, some of the methods that we chose for analysis are Multinomial Logistic Regression, Tree based methods and Support Vector Machines. Using, each of these methods, we were able to identify important predictors that impacted the rating. The results from all the three models were then compared enabling us to draw conclusions about the dataset and the methodologies.

Methods

Dataset Preparation

The dataset used for the analysis is taken from the ggplot2 library of R. It contains data about movies with information such as title, year of release, length, budget, rating, number of votes, MPAA rating and genre. Information about actors and directors for the above movies was extracted from the IMDB database available online.

Actors and Directors were then compared with people's choice of popular actors and directors (Ranker Website). This is a user rating based on number of awards and accolades won by actors and directors and their overall presence on the screen. Based on this list, actors and directors were classified into three categories, with '1' being the most popular, '2' being moderately popular and '3' being unpopular or new to the field. In the same manner as the ratings, these categories were made based on quantiles.

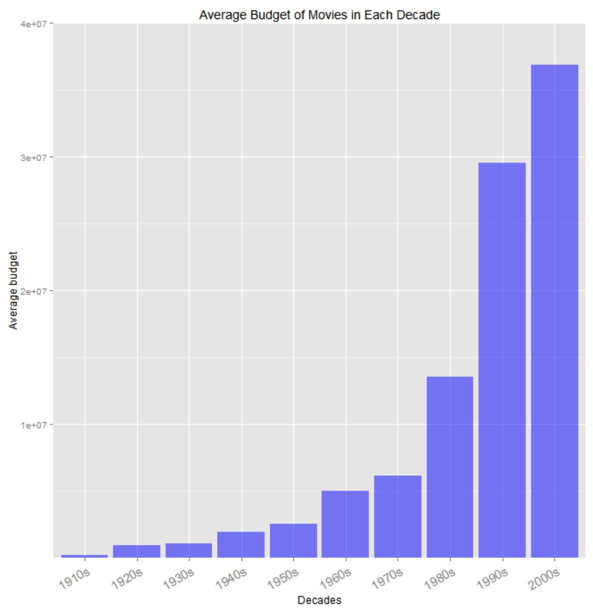
For the purpose of consistency, only full length movies were considered for the analysis. All short movies were removed. Also, all rows with missing values of budget were removed. Thus the final dataset consisted of approximately 2500 rows and 13 predictors.

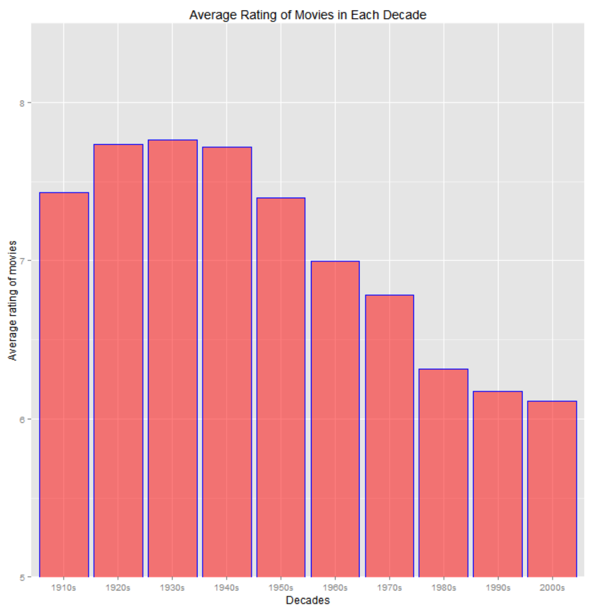
Exploratory Data Analysis

The first step is to understand the data and to see how certain predictors are related with each other and determine if there are any visual patterns between the predictors and the response variable. Below are various plots illustrating the same.

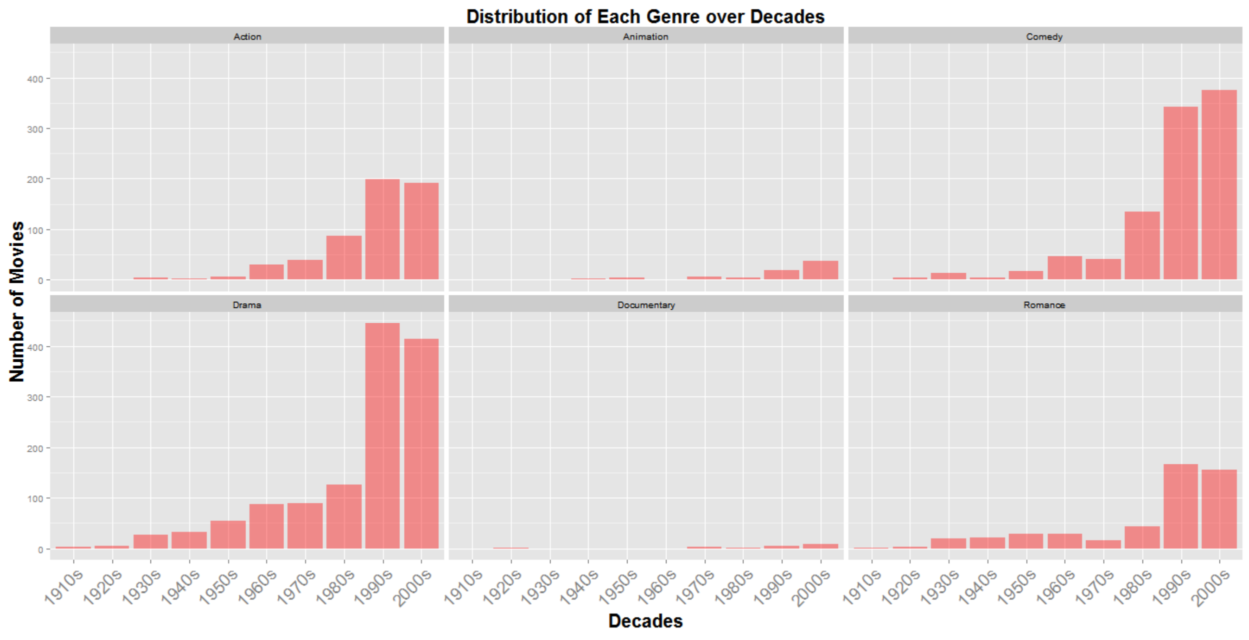
In the below figure, the left histogram shows how the average budget of movies changed over the years. It can be seen that there is a sudden increase in budget from 1980s to 1990s. The right histogram shows how the average ratings of movies changed over time. It shows that movies released in the first half of the century have better average ratings than the movies released in the latter half of the century.

The increase in budget could be attributed to the rising use of technology in film making. Also, the ratings could be attributed to the fact that, people in general are more biased toward older movies and are likely to give it a higher rating. This clearly shows that budget and year of release of the movies are important variables for analysis.

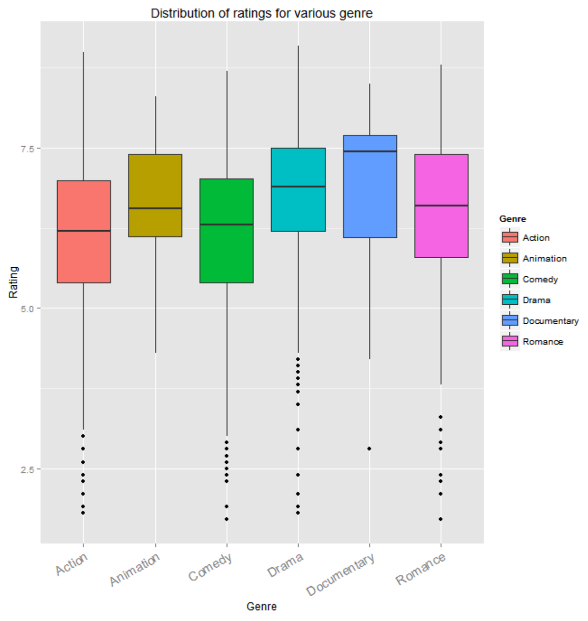


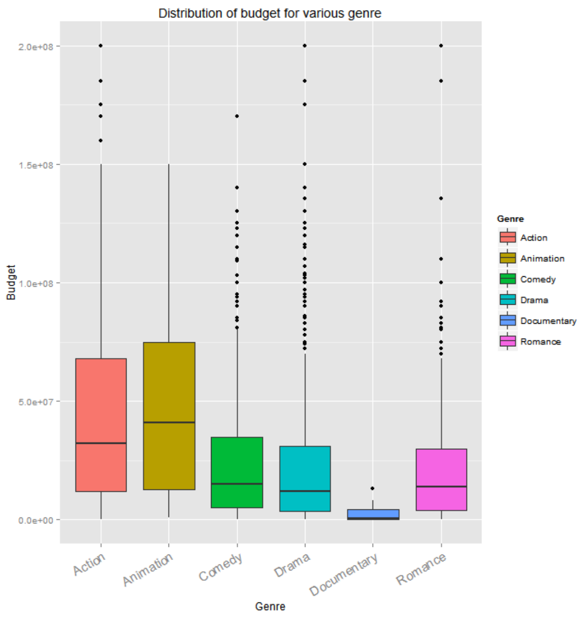


Below is a genre based classification of the total number of movies released in every decade. It can be seen that the highest number of movies are usually made in the drama and comedy genre with the least being in animation and documentary. There is also the possibility that certain movies could belong to more than one genre. For instance, most comedy movies are likely to fall within the drama genre as well. Hence, it is worthwhile to analyze the interactions between genres while building models.

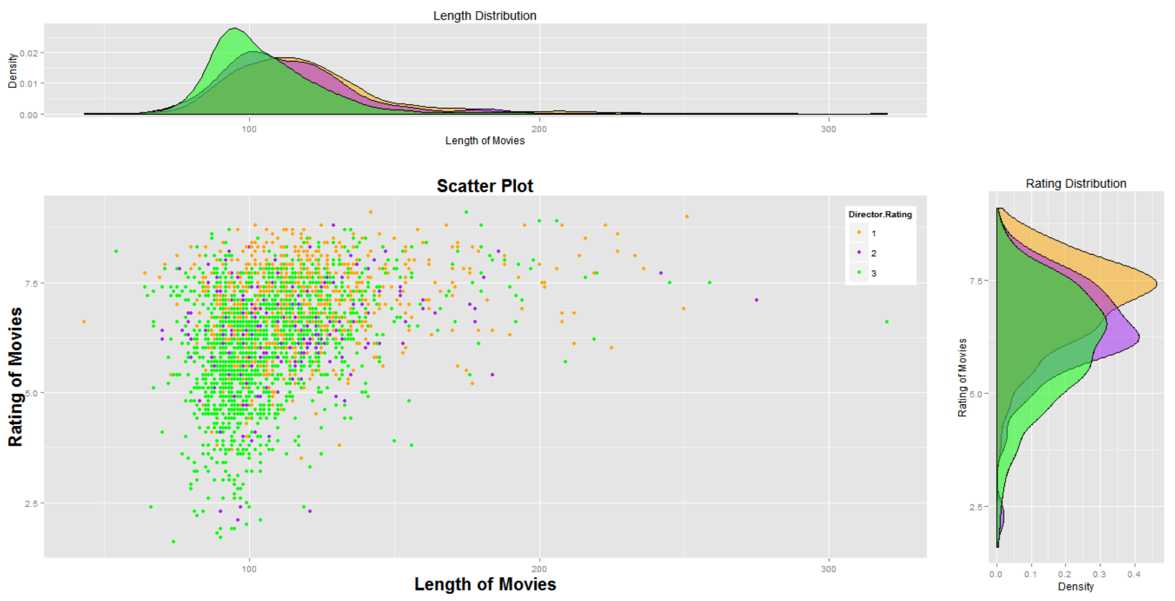


The box plot below is a representation of genre vs rating and genre vs budget. It can be seen that, drama genre tends to have the maximum outliers while documentary having the least. This could be attributed to the number of films made in the respective genres. Drama tends to have the maximum number of movies made in a decade whereas documentary tends to have the least.

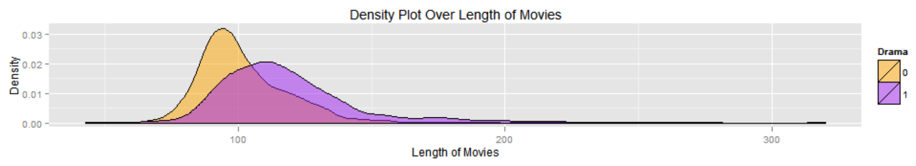


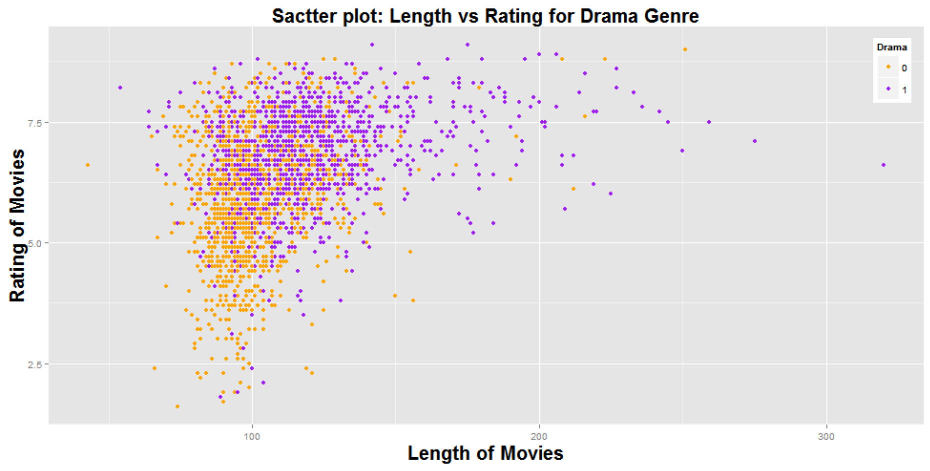


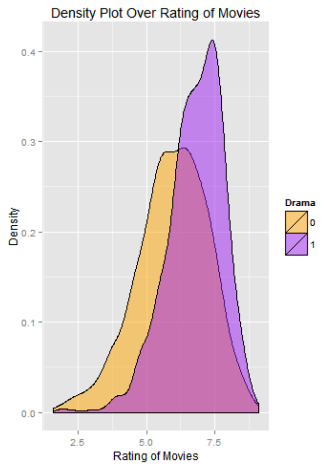
The scatter and density plots depicted below indicate the relation between length of the movies and the ratings. The scatter plot has been divided on the basis of Director Rating. The excellent directors are showcased in orange whereas the good and average directors are showcased in purple and green respectively. It can be observed that, as the length of the movie increases, its rating is likely to be higher. It is also evident that, director rating impacts the rating of the movie directly. A large fraction of excellent directors are in the higher side of the ratings. From here it can be inferred that, length of the movie and the director rating are important predictors directly influencing the ratings.



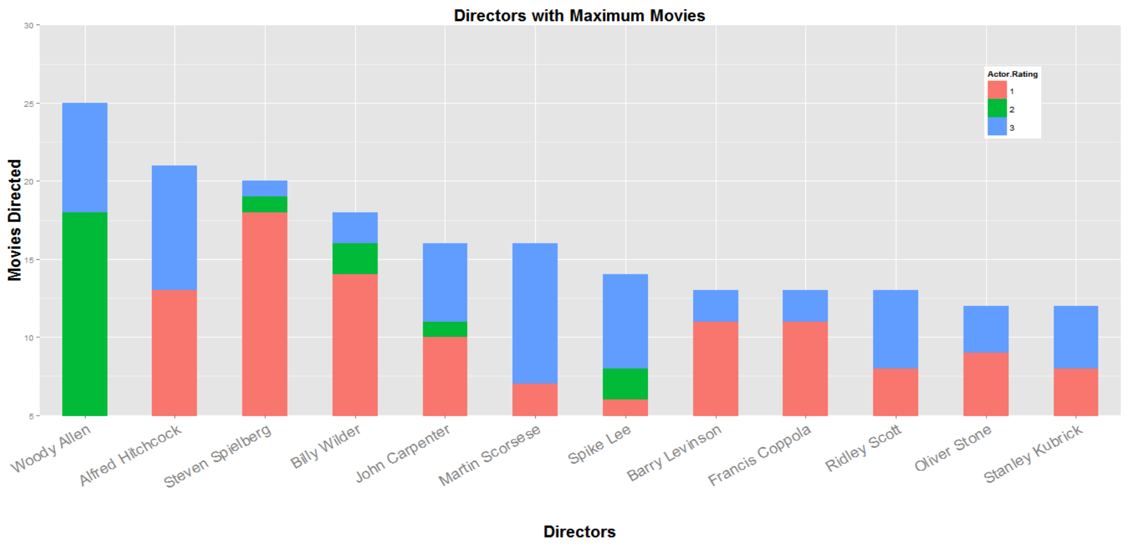
The scatter and density plot below depicts the relation between the length of movies and their rating. Purple indicates drama genre and orange indicates non – drama movies. It can be observed that, longer movies are likely to be in drama genre and tend to have higher ratings. This reiterates the fact that long drama movies tend to do better on an average.







Below is a list of 12 directors in the decreasing order of the number of movies made by them. The three colors represent the type of actors, these directors have worked with. It can be said that, a vast majority of these directors prefer to work with excellent and good actors as opposed to average or new actors. However, from the graph of actors and the type of directors they worked with [1], the same is not true. Implies, actors are more liberal in choosing the directors they work with as opposed to the directors choosing the actors. This indicates that, director rating can prove to be more important than actor rating during prediction.



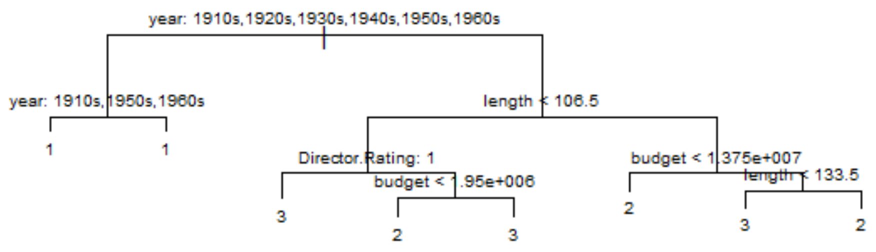
Remark: From the Exploratory analysis it can be said that some of the important predictors that can affect the response are year, length of the movie, director rating and budget.

<u>Tree Based Methods</u>

As this is a classification problem, one of the approaches that was considered was Tree Based Methods. Due to the presence of many predictors and many possible interactions between them, trees were the first choice. Trees tend to perform variable screening or feature selection automatically during the process of classification. Secondly, non-linear relationships between parameters do not affect trees significantly. Lastly, it is easier to interpret the results of decision trees.

Analysis with single trees

From the below plot it can be seen that only certain predictors are considered for the model. The main predictors the tree has chosen for classification are Year of release, Length of the movie, Director rating and Budget.



From the above plot, the following inferences can be drawn.

It can be said that, all movies during and before the 1960's are classified as excellent movies with rating of 1. This is probably due to the following two reasons

1. During this period, the exposure to technology was minimal and hence people's expectations of movies were very less as well. Also, it is a possibility that user ratings for old movies is likely to be higher due to a general bias towards old items.
2. Secondly, it is possible that this particular dataset could be biased towards old movies, having chosen only the good ones from the past.

It can be observed that, movies after the 1960's are classified based on length and budget primarily. If the length of the movie is between 100 and 133 minutes and the budget is less than 13 Million dollars, it is classified as a good movie with a rating of 2.

Similarly, if the length of the movie is between 100 and 133 minutes and the budget is greater than 13 Million dollars, it is classified as an average movie with a rating of 3. However, all movies with a budget greater than 13 Million and length greater than 133 minutes are classified as good movies with a rating of 2.

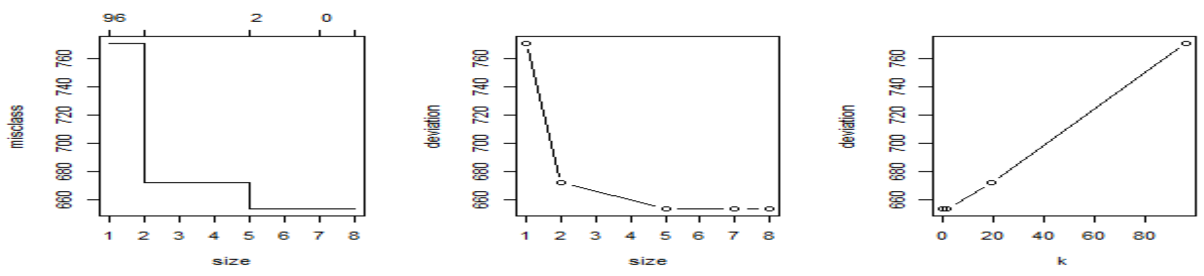
Remark: This is likely due to the fact that, movies within a certain budget and certain time limit, tend to explore ideas fully, whereas short movies with a very high budget tend to leave the audience with unexplored ideas or concepts thus reducing the movie experience. This is corroborated by the fact that movies greater than a certain length and certain budget are classified as good movies.

Accuracy and Error Rate in Model

	Training Dataset	Testing Dataset
Accuracy	59.13	54.86

Remark: It can be seen that, the classification accuracy is very poor. It is just slightly above chance. Also, since the above model uses only a single tree on one particular training dataset, there is a very high variance in the model. Thus to reduce variance and improve the accuracy of prediction, other methods such as Pruning, Bagging and Random Forests were carried out.

Pruning



From the above plots it can be seen that as the size of the tree continues to grow, the misclassification error rate continuously reduces, becoming a constant at size 5. Since, it is better to have a smaller tree with lesser number of splits to reduce variance, pruning is done with size = 5.

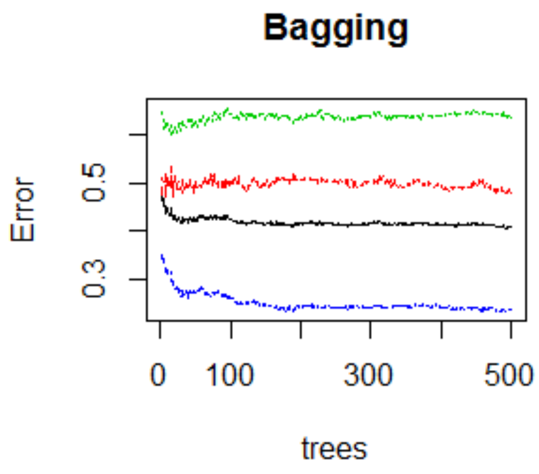
Accuracy and Error rate in Pruned tree

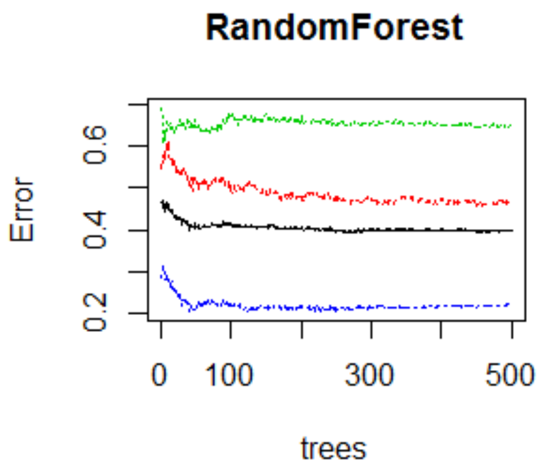
	Training Dataset	Testing Dataset
Accuracy	58.87	55.35

Remark: It can be seen that, there is a slight dip in the training accuracy and an equivalent increase in the testing accuracy. This is mainly because of the reduction in the total number of terminal nodes thus reducing the variance in the model. However, the accuracy is still low and cannot be used for real time prediction.

Bagging and Random Forest

Taking repeated samples from the single training set and generating multiple bootstrapped training sets and fitting a tree model in each of the sets [1]. During prediction, the class with a majority across all samples is the predicted class for a data point. This largely reduces variance and can significantly improve accuracy. The main difference between Bagging and Random Forest lies in the number of predictors randomly sampled. In case of bagging all predictors are used and in random forest square root of the total number of predictors is used. Hence, this reduces the correlation between predictors and tends to decrease variance further.



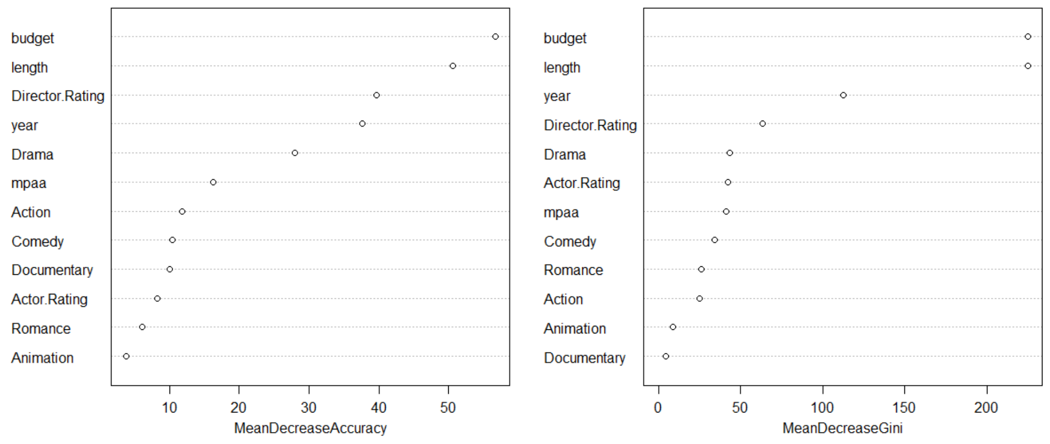


As can be seen from the plot, it is evident that increasing the number of trees is not significantly decreasing the error. However, there is a slight improvement in accuracy as noted below.

	Bagging	RandomForest
Accuracy	58.82%	61.45%

Important Variables

From the below plot, it can be said that the important variables are budget, length of the movie, director rating and year of release. This is true using both accuracy and purity index. Models built using only the top four important variables does not improve results significantly. However, it is very useful for understanding the nature of predictors and the overall dataset.



Model built using top 4 important predictors

	Bagging	Random Forest
Accuracy	59.2%	62.7%

Remark: The model does not significantly improve when only the top four important predictors are used. However, the results don't drop when the rest of the predictors are not used. This shows that these are the four most important predictors driving the model but is not sufficient to adequately predict the ratings. This is likely due to the fact that trees are oversimplifying the prediction process. Hence, to verify, the data was modelled using Logistic regression and SVM.

Multinomial Logistic Regression

A standard logistic regression model is used for 2 class response variable. In case of additional classes, other modelling techniques such as Multinomial Logistic regression is preferred.

Dataset Preprocessing:

Dataset is divided in the ratio 3:2 with three parts for training and two parts for testing. The model is built on the training dataset and is finally tested for accuracy and error rate on the testing dataset.

The package used for this project is 'mlogit' in R. Before 'mlogit' function can be applied the dataset originally in the Wide format needs to be converted to a Long format. The long format consists of each row being repeated the same number of times as there are classes in the response variable. For instance, if there are three classes, each row gets repeated thrice, one row for every class. The output would be the probability of a data point being in a specific class. The class with the highest probability is chosen. The regression coefficients are in the log odds scale.

Model Selection:

For the purpose of building a model with suitable number of predictors, forward selection was the chosen approach. Predictors that gave the highest increase in log likelihood and McFadden R Square values were added one at a time until a stopping criteria was met, in this case the log likelihood value stagnating to a constant.

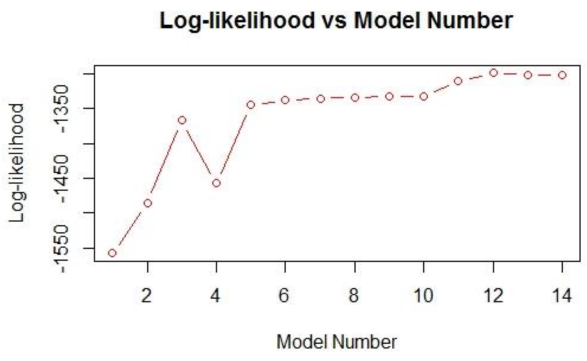
Analysis:

The figure below depicts the graph of log-likelihood vs Model Number. It can be seen that, model 12, gives the highest values of log-likelihood. This model corresponds to the following predictors namely, actor rating, director rating, length, budget, MPAA rating, animation and drama.

From the graph it can also be observed that Model 4 tends to reduce the Log – Likelihood significantly. The model 4 corresponds to Actor rating, director rating and length with length being the newly added term. This indicates the presence of a hidden relationship between length, actor rating and director rating, that yet needs to be explored.

To improve the overall accuracy, possible interactions between the predictors chosen in model 12 were explored. Interactions between director rating and length; budget and MPAA rating; animation and drama improved the overall model in terms of log-likelihood values. Thus the final proposed model consists of actor rating as an independent variable and the above three interactions as other variables.

As expected from model 4, interaction between length and director rating tends to increase log-likelihood whereas length as an independent variable decreases the log likelihood. A possible explanation would be, long movies are usually not preferred unless they are made by good actors and directors.



	Multinomial Logistic Regression
Accuracy (%)	60.07
Error Rate	39.93

Using multinomial logistic regression technique on IMDB data set to predict the rating of any movie, it can be concluded that predictors only affect the movie rating to a certain extent. The model proposed here has an accuracy of about 60% which is reasonable but there is a significant scope for improvement using other classification techniques.

Support Vector Machines

Support vector machines are generalization of a simple and intuitive classifier called the maximal margin classifier, which separates data using a hyper-plane, of the form

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0$, *Where, p is the number of dimensions*

This hyperplane is chosen to be one farthest away from the training observations by computing the (perpendicular) distance from each training observation to a given separating hyper-plane; the smallest such distance is the minimal distance from the observations to the hyperplane, and is known as the margin. Support vectors are the training points, which are closest/on the margin i.e. these points will directly influence the shape of the hyper-plane [3] .The maximal margin classifier is a very natural way to perform classification, if a separating hyperplane exists; else there are techniques like soft margins, kernel trick etc. to help us in that endeavor.

In building our model for the IMDB dataset, which is a classification problem, we noticed complex relationships between the data points and the predictors. This coupled with the fact that our response is multiclass, we decided to apply SVM, to see whether we get improved results compared to the methods we used earlier.

We observed a large number of variables in our dataset, which suggests that we should start with a linear kernel. We then proceed to building the model, on a sampled training dataset with all the predictor variables at once. Since, our response is multiclass, the SVM model will be generated using a 'one-against-one' approach in which k*(k-1)/2, k is the number of classes in the response, binary classifiers are trained and the predicted class is determined by a voting scheme [4].

	Training Dataset	Testing Dataset
Accuracy %	95.3 %	87.8 %

After generating the model we looked at tuning the SVM, to get optimal '*cost*' and '*gamma*' value in order to minimize the error rate across a 10-fold cross validation. And thereafter using, said values were able to generate the optimal SVM model.

	Training Dataset	Testing Dataset
Accuracy % with cost = 0.1	95.7 %	89.12 %

It is difficult to visualize the classification of the data points in an n-dimension hyperspace, but using the predicted values from the model and the confusion matrix generate thereafter, we are able to make reasonable predictions.

Results

From the analysis above it can be said that, Data Exploration, Tree Based Methods and Multinomial Logistic Regression gave concurrent results indicating certain predictors to be more important than others. The predictors are Actor Rating, Director rating, year, length and budget primarily. This implies the model gives the best results with these predictors and the model either deteriorates or does not significantly improve on the addition of other predictors. However in the case of Support Vector Machines, the best results were obtained when all of the predictors were used in the model building process. Also, the results of SVM tends to be significantly better than the results of Tree based Methods and Logistic Regression.

This could be attributed to the fact that, there are certain complex dependencies between predictors that was overlooked by Trees and the Logistic Model. Trees tend to make splits based on the strongest predictors, thus choosing predictors that could be highly correlated among themselves (bagging) or ignoring a complex interaction between two predictors. Also trees tend to oversimplify the classification process. In the logistic model however, it is required to manually understand various possible interactions thus missing out on some vital interaction. SVM's on the other hand does not have both of the above constraints thus emerging as a suitable model for such complex scenarios.

It can be concluded that prediction of movie ratings is possible with a certain degree of accuracy with Support Vector Machines.

Discussion

Performing the analysis and observing the results, we felt to be able to build better models, it is important to participate in the model building process. In the dataset used in the above analysis, we are unsure of how the data was collected and the biases it is likely to have. The model built was not able to capture these fallacies. We also noticed, handling the missing values in a different way could drastically change the results. An interesting future work could involve building models on an independently collected dataset and comparing the results with this. That way the true prediction accuracy can be estimated.

References

1. http://www.rmdk.ca/boosting_forests_bagging.html
2. https://en.wikipedia.org/wiki/Support_vector_machine
3. An Introduction to Statistical Learning with Applications in R by : Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani
4. http://www.inside-r.org/node/57517

Appendix

- Code to extract and clean the dataset from IMDB is in the following Github link below. The final dataset used in the analysis is saved as 'movies.csv' in the same link.

  https://github.com/Qartks89/DataMiningIMDBParsing

- The R code for the rest of the analysis is as below.

```r
########################## Exploratory Data Analysis ######################################

require(plyr)
require(ggplot2)
require(reshape2)
require(gridExtra)


dat <- read.csv("movies.csv", header = T)

dat$Director.Rating <- as.factor(dat$Director.Rating)
dat$Actor.Rating <- as.factor(dat$Actor.Rating)



# Average Budget of Movies in Each Decade

meanbudget <- tapply(dat$budget, dat$year, mean)
meanbudget <- as.data.frame(meanbudget)
meanbudget$years1 <- c("1910s", "1920s", "1930s", "1940s", "1950s", "1960s", "1970s", "1980s", "1990s", "2000s")

plot1 <- ggplot(data=meanbudget, aes(x = years1, y = meanbudget)) +
  geom_histogram(stat="identity", fill = "blue", alpha = 0.5) +
  coord_cartesian(ylim=c(1000, 40000000)) +
  xlab("Decades") + ylab("Average budget") + ggtitle("Average Budget of Movies in Each Decade") +
  theme(axis.text.x = element_text(angle = 30, hjust = 1, size = 14))


# Avearge rating of movies in each deacade

meanrating <- tapply(dat$rating, dat$year, mean)
meanrating <- as.data.frame(meanrating)
meanrating$years1 <- c("1910s", "1920s", "1930s", "1940s", "1950s", "1960s", "1970s", "1980s", "1990s", "2000s")

plot2 <- ggplot(data=meanrating, aes(x = years1, y = meanrating)) +
  geom_histogram(stat = "identity", colour = "blue", fill = "red", alpha = 0.5) +

  coord_cartesian(ylim = c(5, 8.5)) +
  xlab("Decades") + ylab("Average rating of movies") + ggtitle("Average Rating of Movies in Each Decade")


grid.arrange(plot1, plot2, ncol=2, nrow=1)


# Type of genres over the decades

dat$Drama <- as.numeric(dat$Drama)
movie_data_sub <- dat[, c(1,7,8,9,10,11,23,24,25,26,27,28)]
movie_data_sub <- melt(movie_data_sub, c(1,2,3,4,5,6))
names(movie_data_sub)[7] <- c("Genre")
movie_data_sub <- subset(movie_data_sub, value == 1)

```

```r
plot3 <- ggplot(movie_data_sub, aes(year)) + geom_bar(fill = "red", alpha = 0.4) +
  facet_wrap(~ Genre) + theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 18),
                              plot.title = element_text(size = 20, face = "bold"),
                              axis.title.x = element_text(size = 20, face = "bold"),
                              axis.title.y = element_text(size = 20, face = "bold")) +
  xlab("Decades") + ylab("Number of Movies") + ggtitle("Distribution of Each Genre over Decades")
plot3


# Box plot rating and genre


plot4 <- ggplot(data = movie_data_sub, aes(x = Genre, y = rating, fill = Genre)) + geom_boxplot() +
  xlab("Genre") + ylab("Rating") + ggtitle("Distribution of ratings for various genre") +
  theme(axis.text.x = element_text(angle = 30, hjust = 1, size = 14))


# Distribution of budget for various genre boxplot

plot5 <- ggplot(data = movie_data_sub, aes(x = Genre, y = budget, fill = Genre)) +
  geom_boxplot() + xlab("Genre") + ylab("Budget") + ggtitle("Distribution of budget for various genre") +
  theme(axis.text.x = element_text(angle = 30, hjust = 1, size = 14))


grid.arrange(plot4, plot5, ncol=2, nrow=1)



# Scatter plot between length and rating over director rating

empty <- ggplot()+geom_point(aes(1,1), colour="white") +theme(
  plot.background = element_blank(),
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  panel.border = element_blank(),
  panel.background = element_blank(),
  axis.title.x = element_blank(),
  axis.title.y = element_blank(),
  axis.text.x = element_blank(),
  axis.text.y = element_blank(),
  axis.ticks = element_blank())

# Scatterplot of x and y variables

scatter <- ggplot(data = dat,aes(x = length, y = rating)) + geom_point(aes(color = Director.Rating)) +
  scale_color_manual(values = c("orange", "purple", "green")) +
  ggtitle("Scatter Plot") + xlab("Length of Movies") +
  ylab("Rating of Movies") +
         theme(legend.position=c(1,1),legend.justification=c(1,1),
               plot.title = element_text(size = 20, face = "bold"),
               axis.title.x = element_text(size=20, face = "bold"),
               axis.title.y = element_text(size=20, face = "bold"))

       # Marginal density of x - plot on top

       plot_top <- ggplot(data = dat, aes(x = length, fill = Director.Rating)) +
         geom_density(alpha=.5) +
         scale_fill_manual(values = c("orange", "purple", "green")) +
         theme(legend.position = "none") +  ggtitle("Length Distribution") + xlab("Length of Movies") +
         ylab("Density")

       # Marginal density of y - plot on the right

       plot_right <- ggplot(data = dat, aes(x = rating, fill = Director.Rating)) +

         geom_density(alpha=.5) +
         coord_flip() +
         scale_fill_manual(values = c("orange", "purple", "green")) +
         theme(legend.position = "none") + ggtitle("Rating Distribution") + xlab("Rating of Movies") +
         ylab("Density")


       # The above three plots combined

       plot6 <- grid.arrange(plot_top, empty, scatter, plot_right, ncol=2, nrow=2, widths=c(4, 1), heights=c(1.5, 4))



       # Drama v/s All Genre Scatter

       dat$Drama <- as.factor(dat$Drama)
       scatter2 <- ggplot(data = dat,aes(x = length, y = rating)) + geom_point(aes(color = Drama)) +
         scale_color_manual(values = c("orange", "purple")) +
         ggtitle("Sactter plot: Length vs Rating for Drama Genre") +
         xlab("Length of Movies") + ylab("Rating of Movies") +
         theme(legend.position=c(1,1),legend.justification=c(1,1),
               plot.title = element_text(size = 20, face = "bold"),
               axis.title.x = element_text(size=20, face = "bold"),
               axis.title.y = element_text(size=20, face = "bold"))

       # Drama and all other genre length density plot

       plot_top2 <- ggplot(data = dat, aes(x = length, fill = Drama)) + geom_density(alpha=.5) +
         scale_fill_manual(values = c("orange", "purple")) +
         ggtitle("Density Plot Over Length of Movies") + xlab("Length of Movies") + ylab("Density")

       # Drama and all other genre rating density plot

       plot_right2 <- ggplot(data = dat, aes(x = rating, fill = Drama)) + geom_density(alpha=.5) +
         scale_fill_manual(values = c("orange", "purple")) +
         ggtitle("Density Plot Over Rating of Movies") + xlab("Rating of Movies") + ylab("Density")


       # The above three plots combined

       plot7 <- grid.arrange(plot_top2, empty, scatter2, plot_right2, ncol=2, nrow=2, widths=c(4, 1.5), heights=c(1.5, 4))



       # Director names and number of movies

       dat2 <- dat[(dat$Director != " "), ]
       dat2 <- dat2[(dat2$Director != "De"), ]
       dat2 <- dat2[(dat2$Director != " De"), ]
       dat2 <- within(dat2, Director <- factor(Director, levels=names(sort(table(Director), decreasing=TRUE))))

       plot8 <- ggplot(data=dat2, aes(x = Director, fill = Actor.Rating)) + geom_histogram(stat="bin", width = .5) +
         coord_cartesian(xlim=c(0.5, 12.5), ylim=c(5, 30)) +  ggtitle("Directors with Maximum Movies") +
         xlab("Directors") + ylab("Movies Directed") +
         theme(legend.position = c(.9, .8), axis.text.x = element_text(angle = 30, hjust = 1, size=19),
               plot.title = element_text(size = 20, face = "bold"),
               axis.title.x = element_text(size=20, face = "bold"),
               axis.title.y = element_text(size=20, face = "bold"))
       plot8



       ####################################### Trees ##############################################


       #Use final movies dataset
       movies <- read.csv("movies.csv",header = TRUE)

       #Splitting the rating into three categories
       #Rating below 6.5 is category 3
       #Rating below 7.3 and above 6.5 is category 2
       #Rating below 9.1 and above 7.3 is category 1

       movies$rating <- as.factor(ifelse(movies$rating < 6.5, 3,ifelse(movies$rating <= 7.3, 2,ifelse(movies$rating <=9.1,1,NA))))

       #MPAA rating
       #PG 13, NC- 17, PG - category 1
       #R -category 2
       #None - category 3

       movies$mpaa <- as.factor(ifelse(movies$mpaa == "PG-13",1,ifelse(movies$mpaa == "PG",1,ifelse(movies$mpaa == "NC-17",1,ifelse(movies$mpaa =="R",2,3)))))

       #Adding an ID column
       movies$ID <- seq.int(nrow(movies))

       #Converting actor and director ratings to factor variables

       movies$Actor.Rating <- as.factor(movies$Actor.Rating)
       movies$Director.Rating <- as.factor(movies$Director.Rating)

       #Removing all columns that are not predictors
       movies <- movies[,c(29,1,5:9,22:28,10)]

       #Understanding non -factor variables(Length and budget)
       #Length

       hist(movies$length, xlab = "length of movie", col = "grey", prob = TRUE)
       lines(density(movies$length), col = "blue", lwd = 2)
       lines(density(movies$length, adjust = 2),lty = "dotted",col = "darkgreen", lwd = 2)
       plot(density(movies$length), xlab = "length of movie")
       polygon(density(movies$length), col = "Purple",border = "black")

       #Testing for normality - Shapiro -Wilk Test

       shapiro.test(movies$length)

       #Comments - P value is very less. This implies we can reject the null hypothesis
       #and go with the alternate. Hence, it can be said that the sample does not
       #typically follow a normal distribution.

       #QQplots

       qqnorm(movies$length, col = "grey")
       qqline(movies$length, col = 2)
```

```r

#From the above graph, it can be said that, there is a deviation from the
#normal distribution at the tails. Also, since the rest of the sample follows
#a typical order and lies largely within a bell shape(as seen above), the
#lack of normality can be overlooked

#Testing for budget

hist(movies$budget, xlab = "Budget of movie", col = "grey", prob = TRUE)
lines(density(movies$budget), col = "blue", lwd = 2)
lines(density(movies$budget, adjust = 2),lty = "dotted",col = "darkgreen", lwd = 2)
plot(density(movies$budget), xlab = "length of movie")
polygon(density(movies$budget), col = "Purple",border = "black")

#Sahpiro- Wilk normality test

shapiro.test(movies$budget)

#QQ plots
qqnorm(movies$budget, col = "grey")
qqline(movies$budget, col = 2)

#Comments: same as the predictor length

##############################################################################

#Classification Trees

library(tree)

#Sampling the dataset into train and test

set.seed(10)
s <- sample(movies$ID, 1500, replace = FALSE)
train <- movies[s,]
test <- movies[-s,]

movies$ID <- NULL
train$ID <- NULL
test$ID <- NULL
train$Title <- NULL
test$Title <- NULL

#Tree object with all predictors

tree.obj <- tree(train$rating ~ ., data = train)
summary(tree.obj)
plot(tree.obj)
text(tree.obj, pretty = 0, cex = 0.6)

#Misclassification error rate = 43%

#Testing error

pred <- predict(tree.obj, test, type = "class")
t <- table(pred,test$rating)
t

accuracy <- sum(diag(t))/sum(t)
accuracy

#Accuracy = 54%
#error rate = 46%

#Pruning the trees

set.seed(10)
cv.obj <- cv.tree(tree.obj, FUN = prune.misclass)
```

```r
names(cv.obj)
cv.obj
plot(cv.obj)

#Error rates

par(mfrow =c(1,3))
plot(cv.obj$dev ~ cv.obj$size, type = "b", xlab = "size", ylab = "deviation")
plot(cv.obj$dev ~ cv.obj$k, type = "b", xlab = "k", ylab = "deviation")

#Plotting the pruned tree
prune.obj <- prune.misclass(tree.obj, best = 5)
plot(prune.obj)
text(prune.obj, pretty = 0)
summary(prune.obj)

#On test dataset

pred <- predict(prune.obj, test, type = "class")
t <- table(pred,test$rating)
t

accuracy <- sum(diag(t))/sum(t)
accuracy

#Accuracy after pruning - 54%

#Bagging

library(randomForest)
set.seed(10)
bag.tree <- randomForest(train$rating ~ ., data = train, mtry = 12, importance = TRUE)
bag.tree
plot(bag.tree, main = "Bagging")

#prediction using bagging

bag.pred <- predict(bag.tree, newdata = test)
t <- table(bag.pred,test$rating)
t

accuracy <- sum(diag(t))/sum(t)
accuracy

#accuracy after bagging = 59%

#RandomForest

set.seed(10)
random.tree <- randomForest(train$rating ~ ., data = train, mtry = 4, importance = TRUE)
random.tree
plot(random.tree, main = "RandomForest")

#prediction using randomForest

random.pred <- predict(random.tree, newdata = test)
t <- table(random.pred,test$rating)
t

accuracy <- sum(diag(t))/sum(t)
accuracy

#accuracy after random forest = 61%

#Finding important predictors

importance(random.tree)
varImpPlot(random.tree)
```

```r
summary(random.tree)

#Budget > length > year > director rating
#Tree object using important predictors only

tree.imp <- tree(train$rating ~ train$Director.Rating + train$year + train$length + train$budget, data = train)
summary(tree.imp)
plot(tree.imp)
text(tree.imp, pretty = 0)

#Random Forest for important predictors

data1 <- train[,c(2:5,13)]
rand.imp <- randomForest(data1$rating ~ ., data = data1, mtry = 2, importance = TRUE)
rand.imp
plot(rand.imp)

#Accuracy using random forest

data <- test[,c(2:5)]
random.pred.imp <- predict(rand.imp, newdata = data)
t <- table(random.pred.imp,test$rating)
t

accuracy <- sum(diag(t))/sum(t)
accuracy

#accuracy after using important predictors = 58%

#prediction using randomForest

dat <- test[,c(2:5,10,6,1,13)]
random.pred <- predict(random.tree, newdata = dat)
t <- table(random.pred,test$rating)
t

accuracy <- sum(diag(t))/sum(t)
accuracy

#accuracy after random forest = 61%

#Finding important predictors

importance(random.tree)
varImpPlot(random.tree)
summary(random.tree)

#budget > length > year > director.rating > drama > mpaa > actor rating



###################### Multinomial Logistic Regression ####################################


data <- read.csv("movies.csv")

hist(data$rating)      # Histogram of rating values
summary(data$rating)   # Summary of rating column


# Rating values below median(=6.5) go into category-3,values between median and 3rd quartile(=7.3) go into category-2 and values greater than 3rd quartile go into category-1

data$rating[data$rating<6.5] <- 3
data$rating[data$rating>7.3] <- 1
data$rating[data$rating>=6.5 & data$rating<=7.3] <- 2

```

```r
unique(data$rating)           #unique values of rating column

length(data$rating[data$rating==1])         # total= 593
length(data$rating[data$rating==2])         # total= 722
length(data$rating[data$rating==3])         # total= 1222

table(data$rating)

require(mlogit)

data$rating <- as.factor(data$rating)

# The square root of the budget column values has been done to make values small enough to fit in the model

data$budget <- sqrt(data$budget)


# Divide the data set in training and testing data set approximately in 60:40 ratio
set.seed(10)
tr <- sample(1:nrow(data), size = 1500, replace=F)
train <- data[tr,]
test <- data[-tr,]

likelihood <- rep(0,14)

#reshaping the data from wide to long format

mydata <- mlogit.data(train,varying=NULL,choice="rating",shape="wide")
head(mydata)


# Forward variable selection method is applied here to get the most significant variables in the model

model1 <- mlogit(rating ~ 1|Actor.Rating,data=mydata,reflevel="1")
summary(model1)

likelihood[1] <- summary(model1)$logLik[1]

model2 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating,data=mydata,reflevel="1")
summary(model2)

likelihood[2] <- summary(model2)$logLik[1]

model3 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+year,data=mydata,reflevel="1")
summary(model3)

likelihood[3] <- summary(model3)$logLik[1]

model4 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+length,data=mydata,reflevel="1")
summary(model4)

likelihood[4] <- summary(model4)$logLik[1]

model5 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+length+budget,data=mydata,reflevel="1")
summary(model5)

likelihood[5] <- summary(model5)$logLik[1]

model6 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+length+budget+mpaa,data=mydata,reflevel="1")
summary(model6)

likelihood[6] <- summary(model6)$logLik[1]

```

```r
        model7 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+budget+length+mpaa+Action,data=mydata,reflevel="1")
        summary(model7)

        likelihood[7] <- summary(model7)$logLik[1]

        model8 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+budget+length+mpaa+Action+Comedy,data=mydata,reflevel="1")
        summary(model8)

        likelihood[8] <- summary(model8)$logLik[1]

        model9 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+budget+length+mpaa+Action+Comedy+Documentary,data=mydata,reflevel="1")
        summary(model9)

        likelihood[9] <- summary(model9)$logLik[1]

        model10 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+budget+length+budget+mpaa+Action+Comedy+Documentary+Romance,data=mydata,reflevel="1")
        summary(model10)

        likelihood[10] <- summary(model10)$logLik[1]

        model11 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+budget+length+mpaa+Action+Comedy+Documentary+Romance+Animation,data=mydata,reflevel="1")
        summary(model11)

        likelihood[11] <- summary(model11)$logLik[1]

        model12 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating+budget+length+mpaa+Animation+Drama,data=mydata,reflevel="1")
        summary(model12)

        likelihood[12] <- summary(model12)$logLik[1]

        model13 <- mlogit(rating ~ 1|Director.Rating+budget+length+mpaa+Action+Comedy+Documentary+Romance+Animation+Drama,data=mydata,reflevel="1")
        summary(model13)

        likelihood[13] <- summary(model13)$logLik[1]

        model14 <- mlogit(rating ~ 1|Director.Rating+length+budget+mpaa+Action+Comedy+Documentary+Romance+Animation+Drama,data=mydata,reflevel="1")
        summary(model14)

        likelihood[14] <- summary(model14)$logLik[1]

        x <- c(1:14)

        # plot of log-likelihood values of all the models

        plot(x,likelihood,main="Log-likelihood vs Model Number",xlab="Model Number",ylab="Log-likelihood",col="red",type = "b")
        likelihood

        # based on the log-likelihood and R square values we choose 12th model
        # and is selected and analysed further to give better results


        # Trying different combinations of interactive predictors, the final model for predicting the ratings of movie is the below:

        model20 <- mlogit(rating ~ 1|Actor.Rating+Director.Rating*length+budget*mpaa+Animation*Drama,data=mydata,reflevel="1")
        summary(model20)


        # Exponential function of logistic coefficents is calculated below:
```

```r
        # This tells us the log-odds of of being in category 2 versus 1 and 3 versus 1.

        exp(coef(model20))


        #reshaping the test data from wide to long format

        testdata <- mlogit.data(test,varying=NULL,choice="rating",shape="wide")


        # predict function is used to predict the rating on the basis of proposed model

        p <- predict(model20, newdata = testdata)
        p <- data.frame(p)


        l <- length(p$X1)
        m <- rep(0,l)


        # taking out the category from the predicted values which gives the highest probability

        for (i in 1:l){

          m[i] <- which.max(p[i,])

        }


        #table to form 3 by 3 matrix using predicted and the original values from the data

        tab <- table(m,test$rating)
        tab

        # accuracy is calculated

        accuracy <- (tab[1] + tab[5] + tab[9])/sum(tab)
        accuracy_percentage <- accuracy*100
        accuracy_percentage


        # error rate is calculated

        error <- 1-accuracy
        error_percentage <- error*100
        error_percentage



        ############################# Support Vector Machines ######################################


        ### Required Library

        library(e1071)
        library(ROCR)

        ### Importing DataSet

        movies1 <- read.csv("movies.csv")

        movies1$rating <- as.factor(ifelse(movies1$rating < 6.5, 3,ifelse(movies1$rating <= 7.3, 2,ifelse(movies1$rating <=9.1,1,NA))))


        movies1$rating <- as.factor(movies1$rating)
        movies1 <- data.frame(movies1)

        ### Sampling Train and Test Data
        set.seed(15)
        s <- sample(nrow(movies1), 1500)
        train <- movies1[s,]
        test <- movies1[-s,]

        ### Checking Number of Classes in Training and Testing
        table(train$rating)
        table(test$rating)

        ### Training Data Frame
        x = train[,-10]
        y = train[,10]
        dat=data.frame(x=x , y=as.factor(y))


        ### Testing Data Frame
        xt = test[,-10]
        yt = test[,10]
        datt=data.frame(x=xt , y=as.factor(yt))

        ### Building SVM Model using Linear Kernal
        svm.model=svm(y ~ ., data=dat , kernel ="linear", cost=10, gamma = 1)
        summary(svm.model)

        ### Confusion Matrix for Training Data
        train.tab = table(svm.model$fitted , dat$y)
        train.tab
        ### Checking Accuracy
        classAgreement(tab)

        ### Confusion Matrix for Test Data
        svm.pred <- predict(svm.model, datt)
        test.tab=table(pred = svm.pred, true = datt$y)
        ### Checking Accuracy
        classAgreement(test.tab)

        ### Tuning The model for Test data
        tune.out=tune(svm ,y~.,data=datt ,kernel ="linear",
                      ranges =list(cost=c(0.001, 0.01, 0.1, 1,5,10,100,1000))
                      , gamma=c(0.5,1,2,3,4))
        summary(tune.out)
```