IMDB Movie Rating prediction

Badam Vinay Kumar

Guddeti Girish

Abstract

The number of movies produced by the American film studios is growing at an exponential rate, the United States is one of the top most prolific producer of films in the world today. The success rate of the box office is of utmost importance since billions of dollars are invested in the making of each of these movies. In such a scenario, prior knowledge about the success or failure of a particular movie will benefit the production houses since these predictions will give them a fair idea of how to go about with the advertising and campaigning, which itself is an expensive affair altogether. Additionally, being aware of the right time or season to release a particular movie by looking at the overall market will be helpful in in a lot of ways. So, the prediction of the success of a movie is very essential to the film industry. In this project, we give our detailed analysis of the Internet Movie Database (IMDb) and predict the IMDb score for which we make use of a free, user-maintained and online resource pf production details for over 400,000 TV shows and movies. This database contains categorical and numerical information such as IMDb score, director, gross, budget and so on and so forth. The data used in this paper is not freely distributable, but remains copyright to the Internet Movie Database inc. It is used here within the terms of their copying policy. Further distribution of the source data used in this paper may be prohibited.

Introduction

Movie ratings in recent years are influenced by many factors that makes the accurate prediction of ratings for the new movies being released a difficult task. There also have been various semantic analysis techniques to analyze user reviews which were applied to analyze the IMDb movie ratings. None of the studies has succeeded in suggesting a model good enough to be used in the industry. In this project, we attempt to use the IMDb dataset to predict the movie ratings before the movie has been released in the cinema. We have used various regression techniques to predict the rating like linear regression, Support Vector Regression (SVR) and Artificial Neural Networks (ANN). The general design of the prediction system as shown in Figure 1.

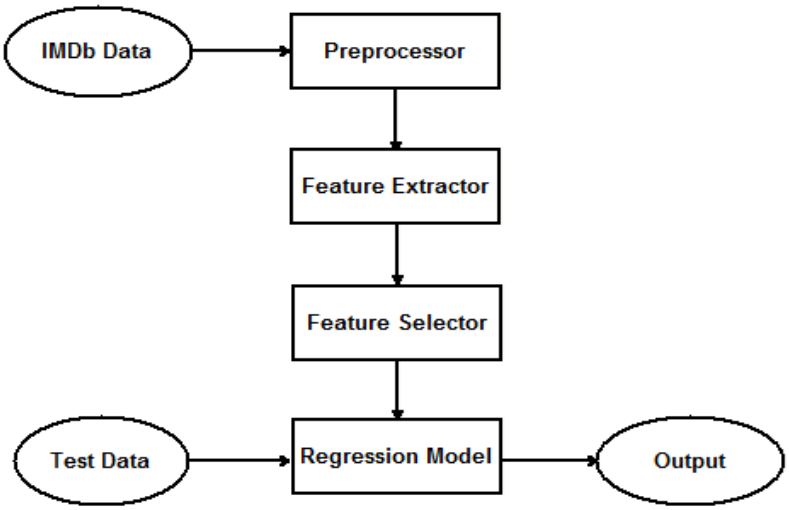


Figure 1: General design of the prediction system

PREVIOUS WORK

"Movie Rating Prediction" by Nick Armstrong and Kevin Yoon (CMU)

IMDB dataset provides extensive information about movies including attributes like directors, actors, number of facebook likes, box office gross. Data pre-processing included dealing with missing data by replacing with mean. Two types of regression techniques are used in predicting movie rating namely linear model tree and kernel based regression. This paper also uses another algorithm called SFS (sequential forward selection), which is a greedy heuristic technique.

SFS

SFS begins with empty set of features and keeps adding the best feature candidates. The candidates are selected based on some goodness measures. This process is repeated until all features are chosen. Best features are selected using P-values, where lower P-values were considered better.

Model tree

A standard decision tree over nominal variables with discrete class is built using m5 algorithm as a model tree. In m5 algorithm, nodes are added based on maximizing the expected error reduction. In general, post-pruning is used to minimize expected errors, whereas in this paper pre-pruning technique was used to determine the set of nodes to be added.

Kernel regression

Kernel regression using heterogeneous distance functions are being applied to classifiers in order to obtain discrete outputs. Distance function called Value Distance Matrix can be applied for discrete classes, but this method could result in loss of information regarding inherent ordering on the output. Hence, the concept of regular kernel regression is combined with normalized heterogeneous functions with different methodologies for parameter selection in order to maximize the performance.

Observation

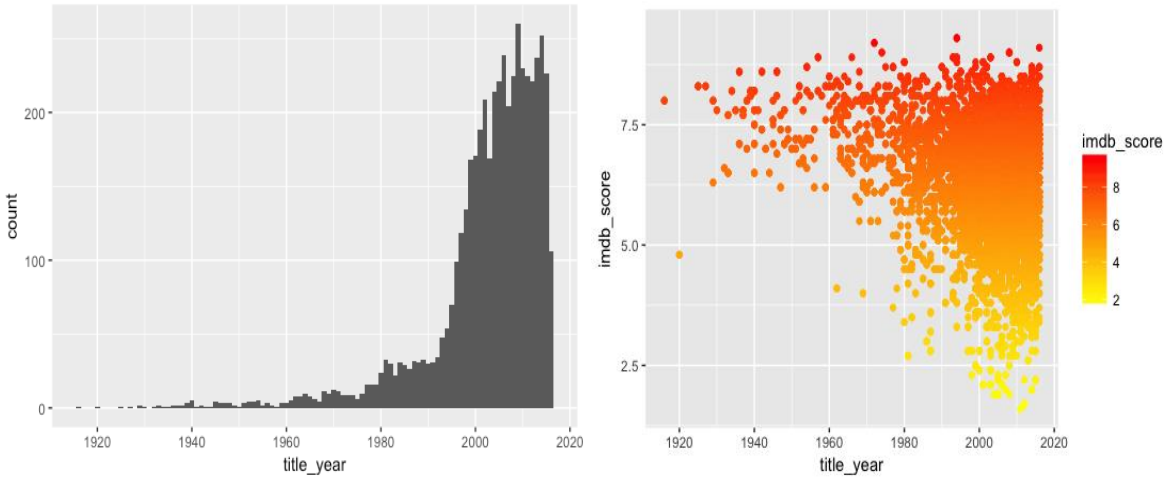
Prediction accuracy obtained from kernel regression has an accuracy of 14.11% deviated from true rating. Model tree performed slightly worser than kernel regression producing an error of 14.4%. Error reduction of about 12.19% was achieved using kernel regression.

Data Pre-processing

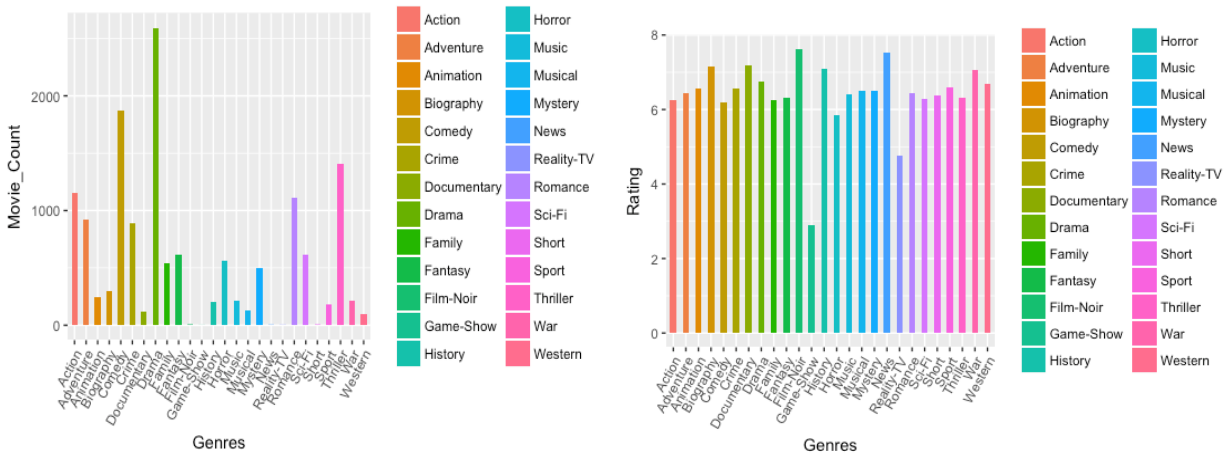
The data obtained by us originated from heterogeneous sources as a result of which they were the data contained many noisy, missing and inconsistent data. To overcome the problem of having missing values, we made use of a method which uses a measure of central tendency for the attribute having missing fields. We have used both mean and median as central tendency post which the duplicated were removed.

Data Visualization

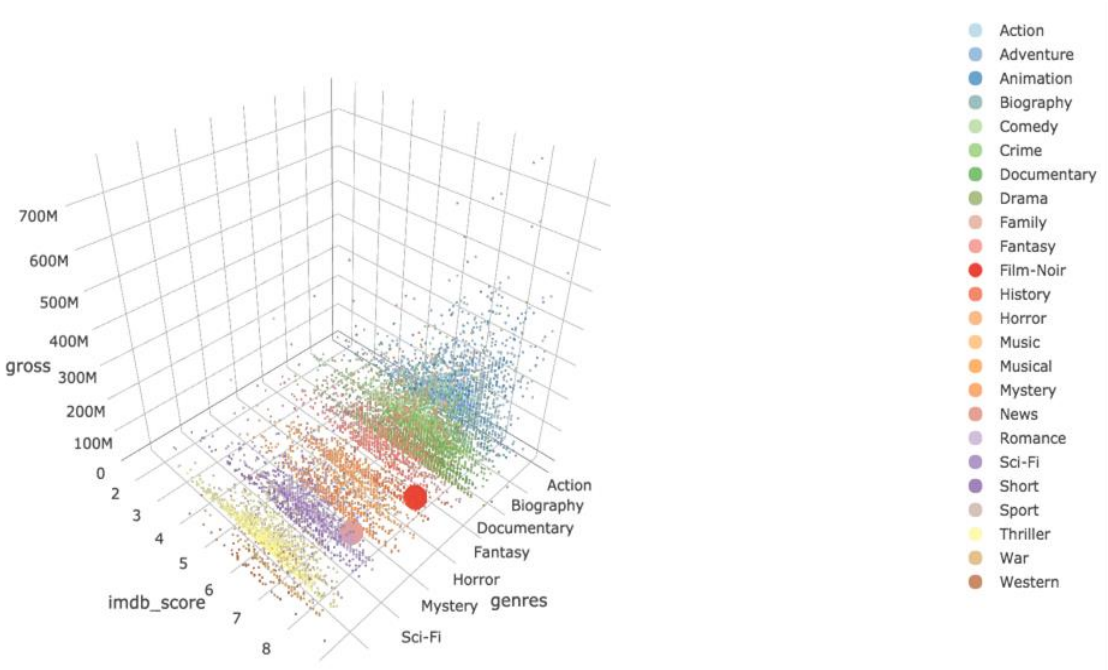
Data visualization is the is the means by which data is summarized organized and communicated using various tools such as diagrams, charts graphs etc. In this project, behavior and patterns of data have been shown by using variety of visualization techniques.



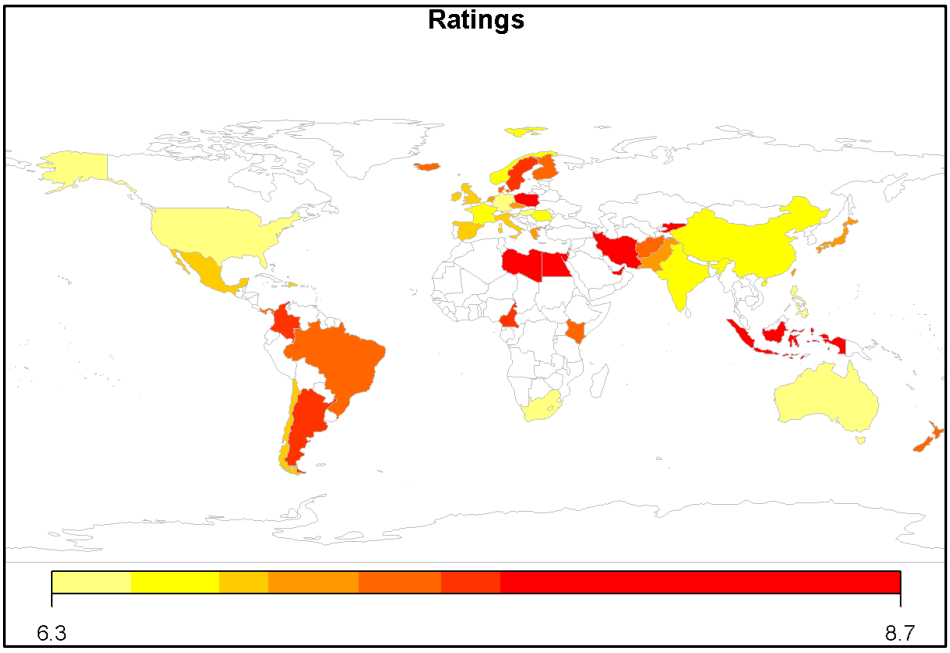
Trend in the number of movies and movie rating over years



Influence of genre on the movie rating



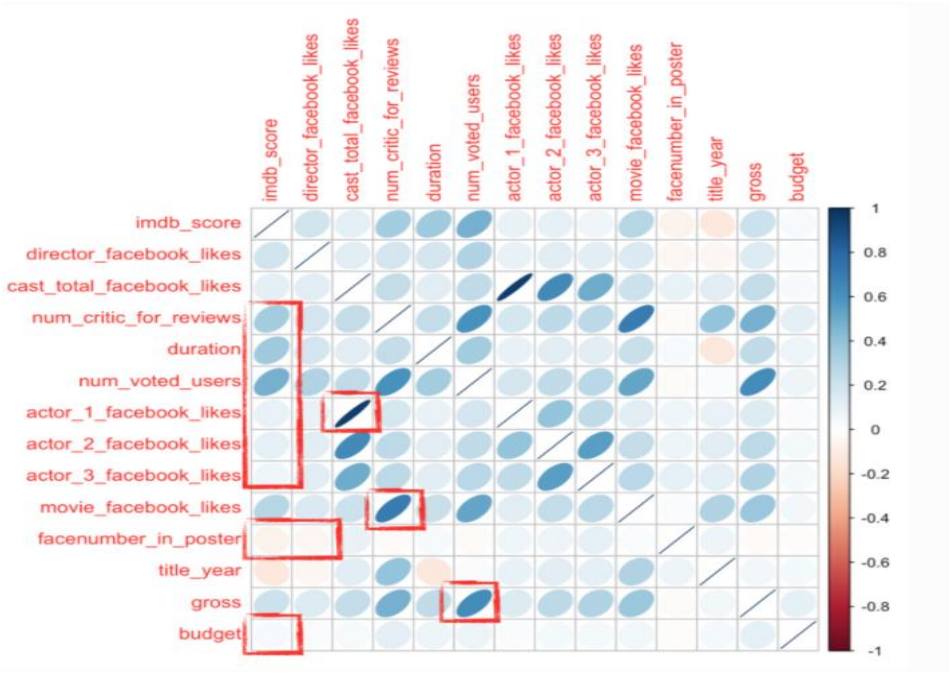
Influence of genre on movie rating and gross



Influence of countries on ratings

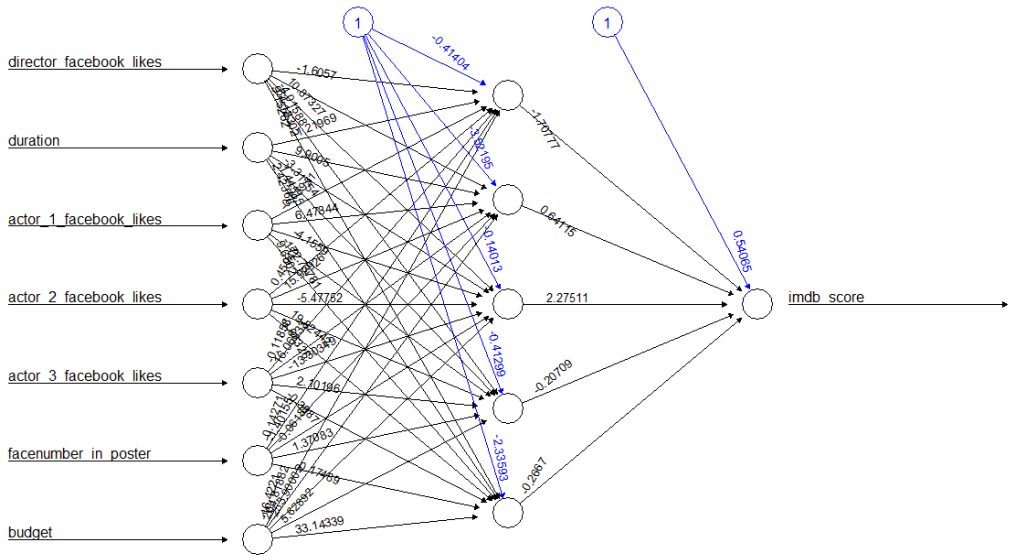
Correlation Analysis

Correlation analysis is a statistical method to evaluate and study the strength of a relationship between two, numerically measured variables. It is done to check if there are possible connections between variables. In this project, correlation analysis is done on imdb score with other numerical variables to find the useful variables for prediction of movie ratings. Small but positive correlation was found with the director_facebook_likes. Small but positive correlation was found with the actor_1, actor_2 and actor_3 facebook_likes. Small but positive correlation was found with duration. Finally, Small but negative correlation was found with facenumber_in_poster and title_year

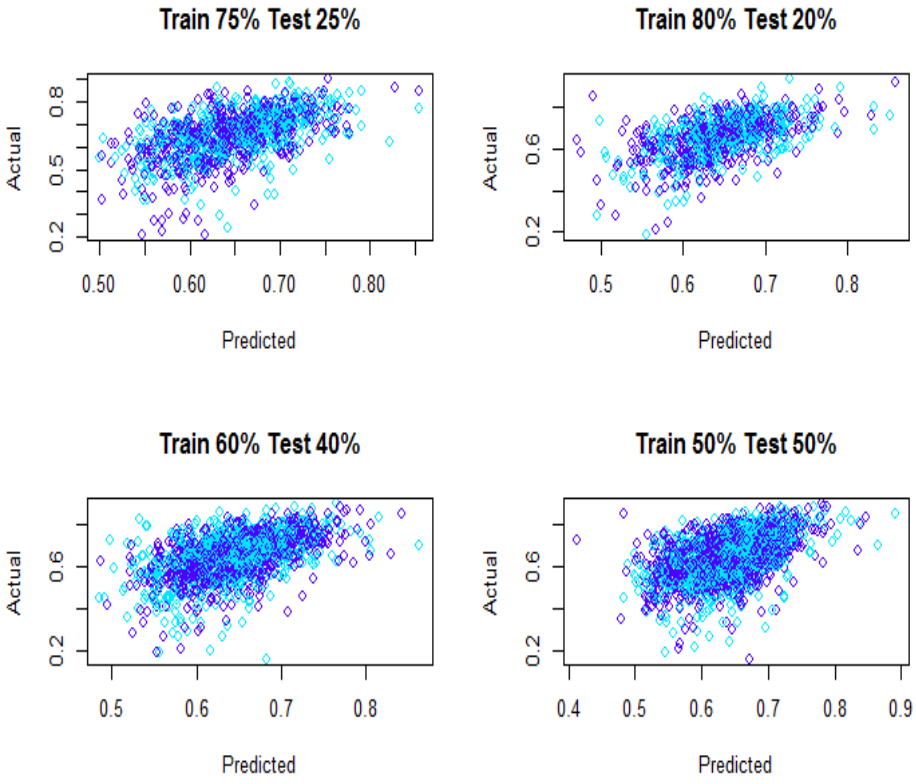


Artificial Neural Network

An artificial neural network is a computational model. The structure of an Artificial Neural Network is affected by the input information that flows through the entire network. By making use of Artificial Neural Networks, complex relationships between the input model and the output model are derived. As a result of this, Artificial Neural Networks can be considered as a nonlinear statistical data modeling tool. There are numerous advantages of using an ANN for prediction. One of the most celebrated advantage is that ANNs can learn from observing the datasets. As a result of this property, ANNs can be used as a random approximation tool. In our project, we have used all the major attributed which contribute the most in the prediction of the IMDb score. These attributes were observed to be the most important ones during the correlational analysis.



Error: 7.810616 Steps: 20924

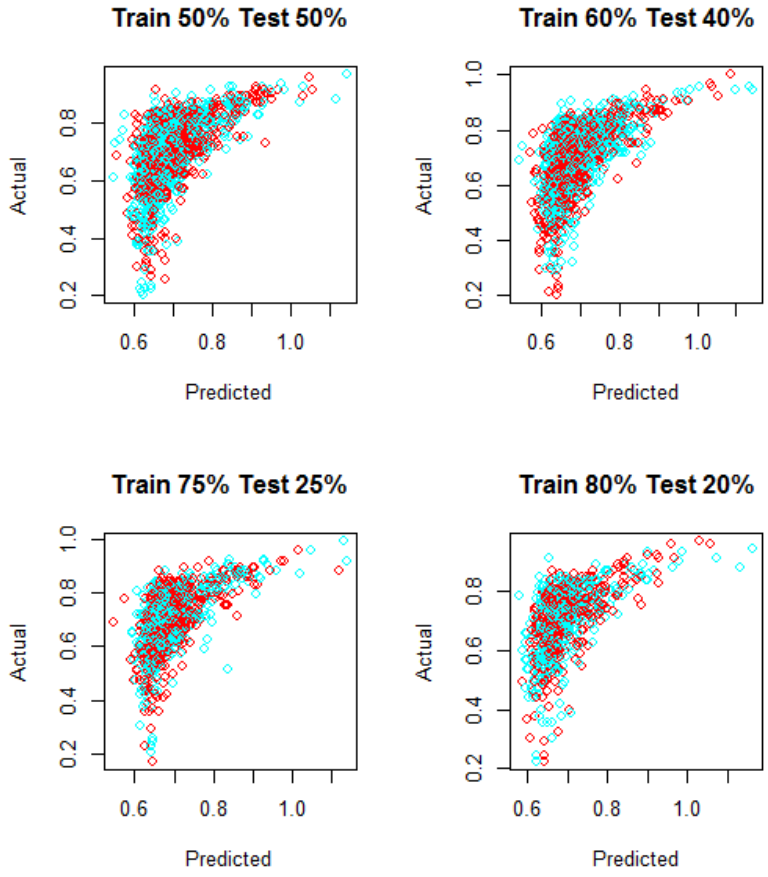


Linear Regression

Linear regression, which is the most basic type of regression is used for predicting movie ratings. Linear regression can be used in studying relationship between two variables namely dependent and independent. In predicting movie ratings, we use imdb_score to be the dependent attribute and set the all numeric attributes to be independent variables. Csv file is read and set of all numeric attributes are extracted. Missing records are removed in order to minimize error and data is scaled as required. Different folds of training and test data sets are used. Model is created for linear regression using training data. Model created is applied to test data to obtain predictions. Root mean square error is estimated. Graph is plotted between actual and predicted data.

Training	Testing	Mean square Error
50	50	0.008502452
60	40	0.009543997
75	25	0.008021885
80	20	0.007549562

Table 3: MSE for different data

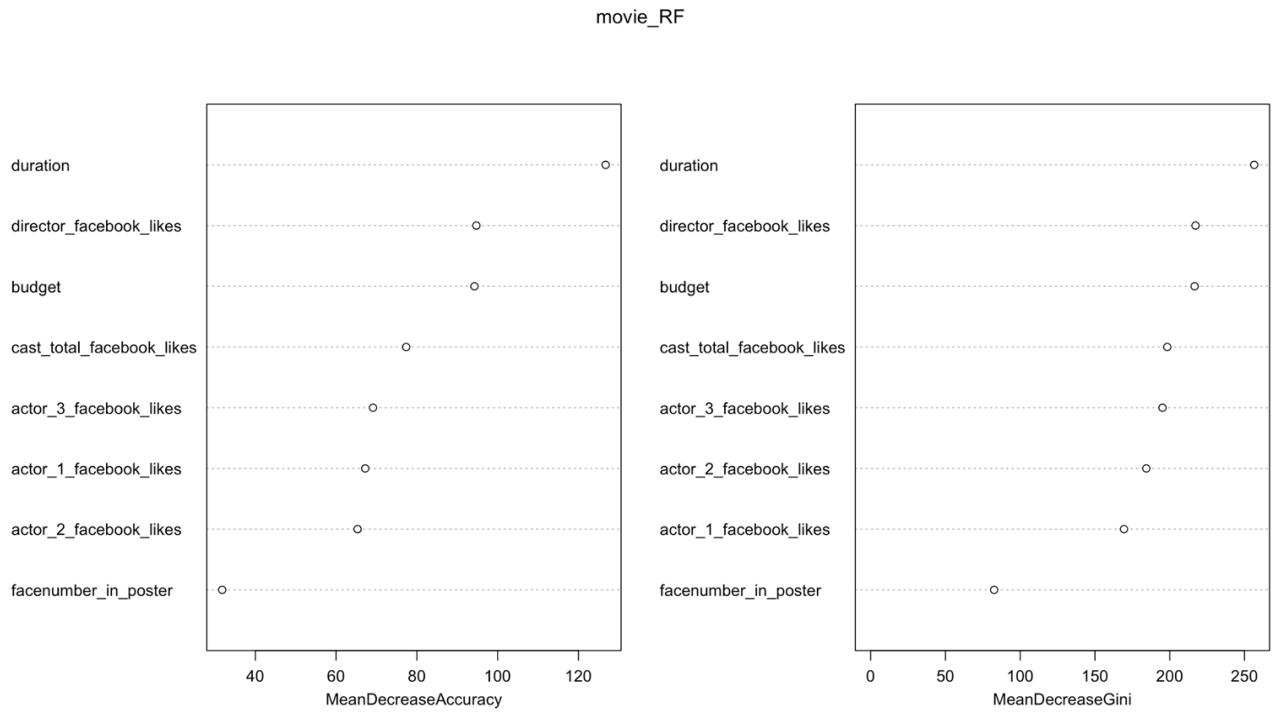


Plot 2: Predicted versus Actual graphs for different train and test data

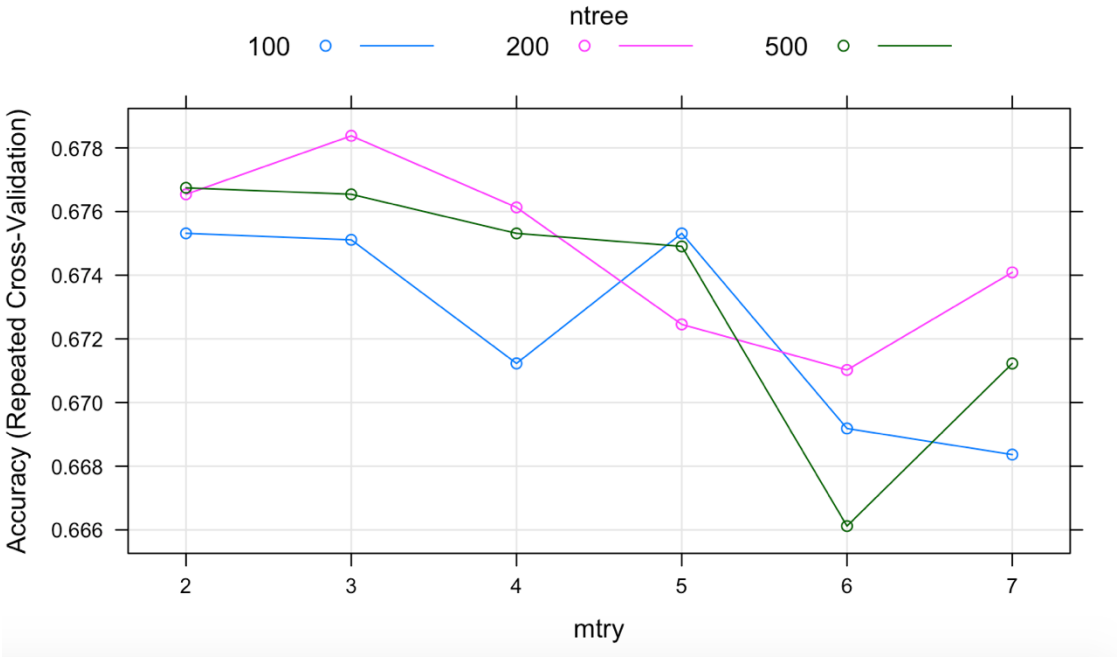
Random Forest Classification

Random forest is an ensemble learning method for classification. This method requires construction of many decision trees at the training time. It was created to improve the performance of decision trees by combining results of individual trees decisions.

Ensemble learning techniques have been developed to avoid over-fitting. For this project, random forest classifier has been used to classify a movie into ratings category by using significant variables such as "imdb_score", "director_facebook_likes", "cast_total_facebook_likes", "actor_1_facebook_likes", "actor_2_facebook_likes", "actor_3_facebook_likes", "movie_facebook_likes", "facenumber_in_poster", "gross", "budget" which were believed to be important variables in classification. Cross-validation has been used to verify the classification by dividing the dataset to 0.632*number of rows for training the model and 0.368*number of rows for testing. Furthermore, Importance of predictor variables are assessed and the multiclass-auc(area under curve) is evaluated and was found to be 0.6283.



Random forest model was tuned with different values number of trees(ntree) and Number of variables randomly sampled as candidates at each split(mtry) to train and the best accuracy found was 0.678.



REFERENCES

[1] https://www.kaggle.com/deepmatrix/imdb-5000-movie-dataset

[2] https://pdfs.semanticscholar.org/cb3d/fb9df1bbfbd7642d3462ccbea8237da70cf2.pdf

[3] http://machinelearningmastery.com/linear-regression-in-r/

[4] https://blog.nycdatascience.com/student-works/machine-learning/movie-rating-prediction/

[5] http://shiny.rstudio.com/tutorial/

Code

Linear Regression

```
getwd()
setwd("D:/PhD/Spring 2017/DM/project/movie_metadata.csv")

#reading csv file
movies <- read.csv("movie_metadata.csv")
#information about type of attributes in movie
str(movies)
# get the set of numeric attributes
numeric_attributes<-sapply(movies,is.numeric)

#attributes containing only numeric data
movies_numeric <- movies[,numeric_attributes]

#removing missing values
movies_missing_removed <- na.omit(movies_numeric)

#movie_data scaled
scaled_movie_data <- data.frame(lapply(movies_missing_removed, function(x) scale(x, center = FALSE, scale = max(x, na.rm = TRUE))))

index_start<- 1:nrow(scaled_movie_data)

#For 50-50
#splitting into training and test data
index_test50 <- sample(index_start, trunc(length(index_start)*0.50))
test_data50 <- scaled_movie_data[index_test50,]
```

```
index_train50 <- sample(index_start, trunc(length(index_start)*0.50))

train_data50 <- scaled_movie_data[index_train50,]



#create model using linear regression

linear_model <-glm(imdb_score ~., data = train_data50)

#applying the model created on test data to obtain predictions
linear_predictions1 <- predict(linear_model, test_data50)

#plot(linear_predictions,test_data50$imdb_score,col=c("red","green"),
xlab="Predicted",ylab="Actual")

#root mean square error
mean((test_data50$imdb_score - linear_predictions)^2)



#For 60-40
#splitting into training and test data
index_test40 <- sample(index_start, trunc(length(index_start)*0.40))
test_data40 <- scaled_movie_data[index_test40,]

index_train60 <- sample(index_start, trunc(length(index_start)*0.60))

train_data60 <- scaled_movie_data[index_train60,]
```

```
#create model using linear regression

linear_model <-glm(imdb_score ~., data = train_data60)

#applying the model created on test data to obtain predictions
linear_predictions2 <- predict(linear_model, test_data40)

#plot(linear_predictions,test_data40$imdb_score,col=c("red","green"),
xlab="Predicted",ylab="Actual")

#root mean square error
mean((test_data40$imdb_score - linear_predictions)^2)

#For 75-25
#splitting into training and test data
index_test25 <- sample(index_start, trunc(length(index_start)*0.25))
test_data25 <- scaled_movie_data[index_test25,]

index_train75 <- sample(index_start, trunc(length(index_start)*0.75))

train_data75 <- scaled_movie_data[index_train75,]


#create model using linear regression

linear_model <-glm(imdb_score ~., data = train_data75)

#applying the model created on test data to obtain predictions
linear_predictions3 <- predict(linear_model, test_data25)
```

```
#plot(linear_predictions,test_data25$imdb_score,col=c("red","green"),
xlab="Predicted",ylab="Actual")

#root mean square error
mean((test_data25$imdb_score - linear_predictions)^2)


#For 80-20
#splitting into training and test data
index_test20 <- sample(index_start, trunc(length(index_start)*0.20))
test_data20 <- scaled_movie_data[index_test20,]

index_train80 <- sample(index_start, trunc(length(index_start)*0.80))

train_data80 <- scaled_movie_data[index_train80,]


#create model using linear regression

linear_model <-glm(imdb_score ~., data = train_data80)

#applying the model created on test data to obtain predictions
linear_predictions4 <- predict(linear_model, test_data20)

#plot(linear_predictions,test_data20$imdb_score,col=c("red","green"),
xlab="Predicted",ylab="Actual")

#root mean square error
```

```
mean((test_data20$imdb_score - linear_predictions)^2)


par(mfrow=c(2,2))
plot(linear_predictions1,test_data50$imdb_score,col=rainbow(2),
xlab="Predicted",ylab="Actual", main="Test data 50")
plot(linear_predictions2,test_data40$imdb_score,col=rainbow(2),
xlab="Predicted",ylab="Actual", main= "Test data 40")
plot(linear_predictions3,test_data25$imdb_score,col=rainbow(2),
xlab="Predicted",ylab="Actual", main="Test data 25")
plot(linear_predictions4,test_data20$imdb_score,col=rainbow(2),
xlab="Predicted",ylab="Actual",main="Test data 20")
```

Random Forest

```
library(randomForest)
library(pROC)

library(mlbench)
library(caret)
movies <- read_csv("~/Downloads/Datamining/movie_rating_prediction/movie_metadata.csv")
set.seed(0)
movies_with_good_variables = movies[, c("imdb_score",
                                        "director_facebook_likes",
                                        "cast_total_facebook_likes",
                                        "actor_1_facebook_likes",
                                        "actor_2_facebook_likes",
                                        "actor_3_facebook_likes",
                                        "movie_facebook_likes",
                                        "facenumber_in_poster",
                                        "gross",
                                        "budget")]
```

```
mvs = na.omit(movies_with_good_variables)

#finding mean imdb score

mean_rating <- mean(mvs$imdb_score)

#discretization of imdb score using mean

#if the score is less than "floor of the mean -1" then rating is poor else if score  greater

#than ceiling of mean rating is good. else rating is average

mvs$rating <- ifelse(mvs$imdb_score < (floor(mean_rating)-1),'poor',ifelse(mvs$imdb_score > ceiling(mean_rating),'good', 'average'))

#convert to factors

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

#66.32% of data is recommended to be ideal training size for random forest hence 36.8% for test


index = sample(1:nrow(mvs), size=0.368*nrow( mvs))

test = mvs[index,]

train = mvs[-index,]

#performing random forest on training set. Not including imdb score now

movie.rf <- randomForest(rating ~. -imdb_score, train, replace=TRUE,na.action=na.omit , mtry = 5, ntree= 100)


#first check on training set if prediction is happening correctly

movie_train_prediction <- predict(movie.rf, train)

table(train$rating, movie_train_prediction)

#above table will show if prediction model is correct on training data

#perform prediction on test data

movie_test_prediction <- predict(movie.rf, test)

#show the confusion matrix of test data

table(test$rating, movie_test_prediction)

#show weightage of each attribute in classifier model to know the significance of variable

#show accuracy
```

```
control <- trainControl(method="repeatedcv", number=2, repeats=2, search="grid")

set.seed(7)

seed <- 7

metric <- "Accuracy"

set.seed(seed)

#mtry <- sqrt(ncol(train))

customRF <- list(type = "Classification", library = "randomForest", loop = NULL)

customRF$parameters <- data.frame(parameter = c("mtry", "ntree"), class = rep("numeric", 2), label = c("mtry", "ntree"))

customRF$grid <- function(x, y, len = NULL, search = "grid") {}

customRF$fit <- function(x, y, wts, param, lev, last, weights, classProbs, ...) {

  randomForest(x, y, mtry = param$mtry, ntree=param$ntree, ...)

}

customRF$predict <- function(modelFit, newdata, preProc = NULL, submodels = NULL)

  predict(modelFit, newdata)

customRF$prob <- function(modelFit, newdata, preProc = NULL, submodels = NULL)

  predict(modelFit, newdata, type = "prob")

customRF$sort <- function(x) x[order(x[,1]),]

customRF$levels <- function(x) x$classes

tunegrid <- expand.grid(.mtry=c(2:4), .ntree=c(100, 200))

rf_gridsearch <- train(rating ~. -imdb_score, data=train, method=customRF, metric=metric, tuneGrid=tunegrid, trControl=control)

print(rf_gridsearch)

pred<- predict(rf_gridsearch, train)

table(pred, train$rating)

pred<- predict(rf_gridsearch, test)

table(pred, test$rating)

rightPred <- pred == test$rating

accuracy <- sum(rightPred)/nrow(test)
```

```
plot(rf_gridsearch)

multiclass.roc(as.numeric(test$rating), as.numeric(pred))
```

Analysis:

After testing the dataset with multiple algorithms, we got a Random Forest accuracy of 67.8 and Linear egression of 95% Overall, from among those algorithms Linear Regression with the highest accuracy of 95%. Random Forest gave poor accuracy among all models because most of the data is linear in this dataset and this dataset is distributed equally