Movie Industry Insights and Recommender System

Machine Learning Project Report

Data 311

Pardeep Rathore

Wasek Habib

Dhruv Virani

Vaibhav R. Kumar

Devon McNeil

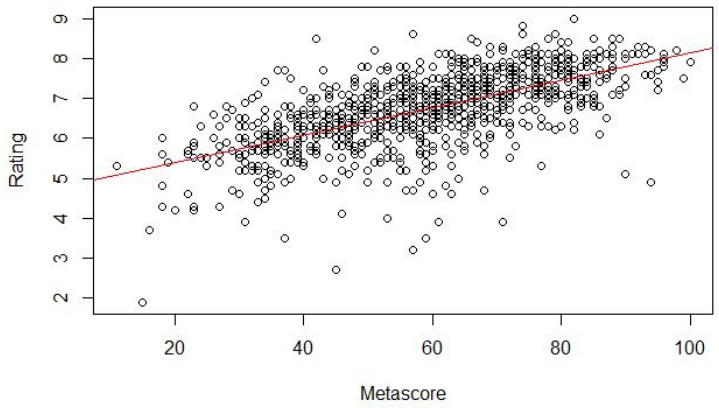
Content

Introduction

The primary purpose of the analysis report is to create meaningful insights using predictive analytics and machine learning techniques for the film industry so the studios can make their next big hit. Another objective is to create a recommender system and predict user ratings for movies they have not rated yet based on their preferences.

Data:

Two datasets have been used. The first data set is one of the many sets of data on Kaggle.com and is titled *IMDB data from 2006 to 2016*. The data set contains information on 1000 movies and the variables from the data set that was primarily used for analysis are Rank, Genre, Year, Runtime(minutes), Rating, Votes, Revenue(millions), and Metascore. A new variable NewGenre was also created which contains the first genre listed in each row of the Genre column as some genre variables have multiple genres in them. For a more in-depth look at the variables in the data set please refer to the readme file submitted alongside the report. After cleaning the data set removing any empty value there were 837 useable rows of data left. An initial simple linear regression was done on the data set variables Rating and Metascore using Rating as the response and Metascore as the predictor. This was done so that the data set could be checked if it was suitable for analysis.



The second dataset has been obtained from movielens.org website that contains ratings of 100,000 movies and only used for the recommender engine since the first dataset lack of user information. `userId,movieId, and rating` are used from the ratings.csv file. Furthermore, `movieId and title` are used from the movies.csv file. Please refer to the README.txt file for detailed explanation.

Classification Tree:

The classification tree is a type of decision tree (a supervised learning method) and is one of the most popular machine learning algorithms. A decision-tree model in which the target variable can produce response variables with a discrete set of values or categories is called classification tree. Idea is to move towards the targeted outcome while making several decisions step-by-step using the input data.

Analysis:

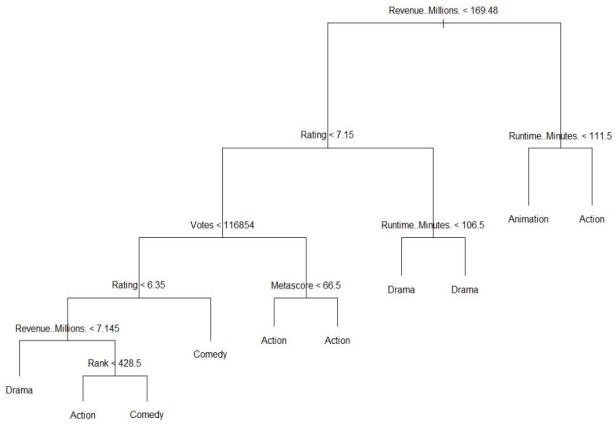
The classification tree is used to predict qualitative response; therefore, it is used to predict genre as the given set of predictor variables. The decision tree algorithm runs on the dataset.

```
Number of terminal nodes:  10
Residual mean deviance:  3.094 = 2562 / 828
Misclassification error rate: 0.5609 = 470 / 838
```

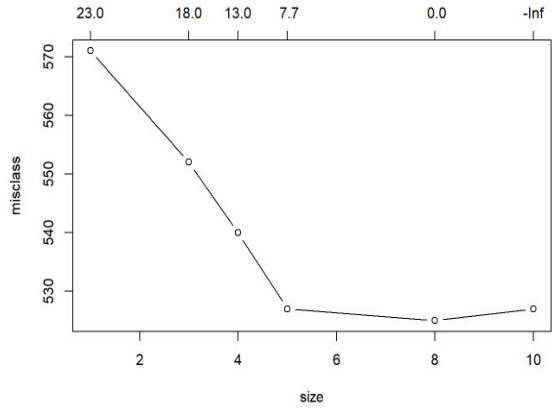
It is observed that the training error rate is 56.1%. For classification trees, the deviance reported in the output of summary() is given by

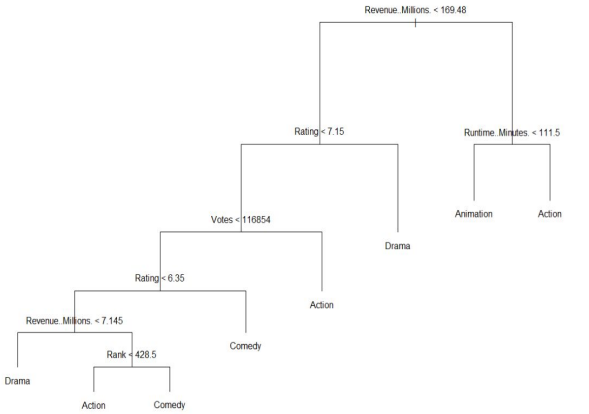
$$-2\sum_{m}\sum_{k} n_{mk}\log\hat{p}_{mk},$$

where nmk is the number of observations in the mth terminal node that belong to the kth class. The residual mean deviance reported is simply the deviance divided by n−|T0|, which in this case is 838−10 = 828.



After plotting the tree, it is observed that the tree does not provide a suitable conclusion in some cases. The left-hand side is the unpruned tree. Pruning technique is used to fix the problem. Pruning is a technique that reduces the complexity of final classifier and improves accuracy by reduction of overfitting.





Evidently, the resulted output is expected after pruning.

```
tree_pre <- predict(bocl, test.data, type = "class")

Mean <-mean(tree_pre != testing_NewGenre)
MeanTest <- paste("Classification tree misclassification calculation with testing data. ", round(Mean,4))
MeanTest
```

```
## [1] "Classification tree misclassification calculation with testing data.  0.5728"
```

Now 57.3% of the test observations are correctly classified, which is almost agreed to training data(56.1%).

Random Forest

The main con of trees is that they are unstable because of the high variance. The small changes in data can result in a completely different tree. The reason could be for instability is that if a split changes, it affects all other splits as well. Therefore, Random forest is used to stabilize the tree-based approach.

Random forest is a supervised learning algorithm, and it can be used for both classification and regression. The idea is to decorrelate the several trees which are generated by the different bootstrapped samples from training data, and then we simply reduce the variance in the trees by averaging them.

Analysis:

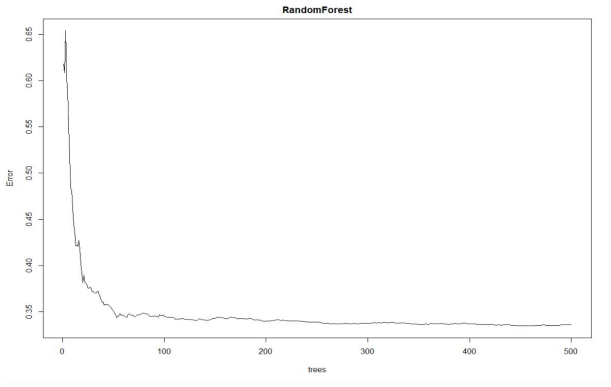
Random forests randomly removed predictors from consideration at each binary split. This means trees have (individually) less information to grow; therefore, this leads to a larger variety of trees and less correlation across the trees and more stability the better predictions.

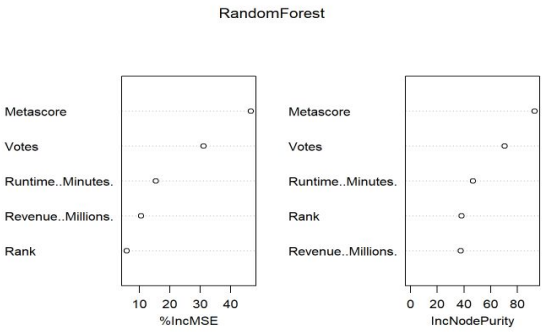
```
Call:
 randomForest(formula = Rating ~ Votes + Runtime..Minutes. + Revenue..Millions. +      Metascore + Rank, data = train.data, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 0.335725
                    % Var explained: 54.98
[1] "Predicting Random Forest with testing dataset  0.355486"
```

The goal was to use a random forest to predict rating based on the 500 trees that have the largest variance in the training set. The observations are divided randomly into training and testing sets and applied random forest to train data for m = p/3 (p/3 by default in case of regression tree). The

error rate of a single tree is 33.6 %, and test data has the approximately same error rate of 35.5%.





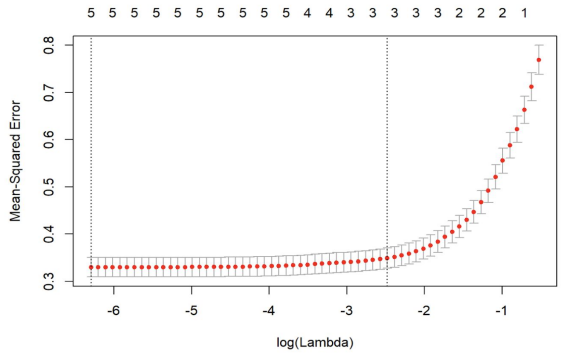
In the case of regression trees, the node impurity is measured by the training RSS, The results indicate that across all of the trees considered in the random forest, the metascore is the most important variable and the votes are the two most important variable.

Lasso

Lasso regression is a linear regression which makes use of shrinkage and feature selection. It is also helpful to solve the multicollinearity problem by suggesting important variables according to minimum CV MSE and 1 Standard Error. Shrinkage means to compress the data to a point, such as the mean.

Analysis:

To perform Lasso, a library known as 'glmnet' is imported. Lasso is used because it is very efficient in dealing with data groups containing high correlated variable by selecting one variable from the group and ignoring other variables.



According to the graph, it can be observed that the mean squared error increases as

the value of log(lambda) decrease. Furthermore, on the left-hand side dashed line is minimum CV MSE and the right-hand side is CV within one standard error of minimum. According to the minimum CV MSE, it is not possible to get rid of any of the predictors.

However, for within one standard error of minimum, for every unit moved rightwards there is a big increase on the mean squared error. Therefore, it can be concluded that the best results are towards the leftmost side where the mean squared error is the least.

Multiple Linear Regression (MLR)

MLR differs from Linear Regression (LR) where there are more than one predictor variables for MLR. For the model used, Rating is our response variable with Votes, Runtime, Revenue and Metascore as our predictor variables.

Analysis:

In order to determine the important predictor variable, the backward selection technique is used on the MLR model. Initially, all predictors were included in the model, a significance level of 0.05 is used and the model is initially fitted with all possible predictor variables. The predictor variable with the highest p-value is removed. Since the p-value of Rank is greatest (i.e. p-value of 0.215), the Rank has been removed for the second run.

```
##
## Call:
## lm(formula = Rating ~ Votes + Runtime..Minutes. + Revenue..Millions. +
##     Metascore)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3659 -0.2844  0.0301  0.3440  1.8789
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         4.126e+00  1.395e-01  29.587  < 2e-16 ***
## Votes               1.656e-06  1.449e-07  11.432  < 2e-16 ***
## Runtime..Minutes.   7.009e-03  1.176e-03   5.960 3.73e-09 ***
## Revenue..Millions. -1.118e-03  2.470e-04  -4.526 6.89e-06 ***
## Metascore           2.785e-02  1.251e-03  22.268  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5724 on 833 degrees of freedom
##   (162 observations deleted due to missingness)
## Multiple R-squared:  0.5767, Adjusted R-squared:  0.5747
## F-statistic: 283.7 on 4 and 833 DF,  p-value: < 2.2e-16
```

On the second run, the MLR model shows that four of our five predictor variables are significant namely Votes, Runtime, Revenue and Metascore. The multiple R squared values is about 57.2%.

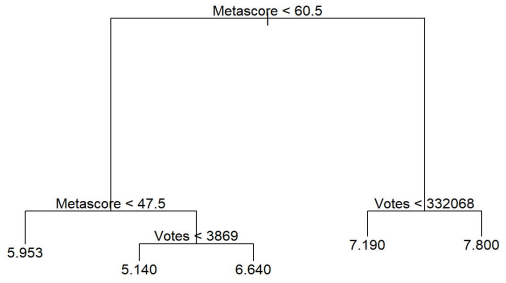
This means the predictor variables are contributing more than half to the overall variability.

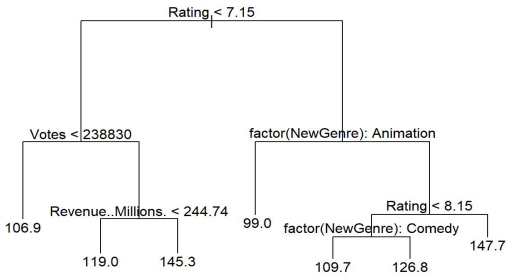
Regression tree:

A regression tree is a type of decision tree that splits the data into R1, . . . , RJ regions using recursive binary splitting to minimize the RSS in each region. It then uses regression to predict the mean response for the observations that fall into each region from the training set.

Analysis:

A regression tree was run once using each of the seven variables Rating, Rank, NewGenre, Runtime(Minutes), Votes, Revenue(Millions), and Metascore as the response variable. Based on the types of the variable it was quickly determined that no good results would be gained from the regression tree using NewGenre as its response variable is categorical. The calculated MSE of each of the remaining variables was then used to calculate the standard deviation each regression tree for the corresponding response variable. The standard deviation was then compared to the value range of the response variable to determine if the regression tree was adequate at predicting the response variable.





The trees shown are the pruned trees from the regression trees that were determined to adequately predict their response. From top right, they are for rating, runtime, and metascore.

Out of the trees that were determined to be good it is interesting that runtime is able to be predicted and that its prediction is based on both categorical and numeric data.

Boosting:

Boosting is a method for improving the predictions resulting from a decision tree, in this case, a regression tree. It does this by growing a number of trees n sequentially using information from the previously grown trees. By using this approach boosting learns slowly allowing the model to avoid potentially overfitting the data, which can happen when a single large decision tree is fitted to the data.

Analysis:

As with the regression tree model the boosting model was run once using each of the seven variables Rating, Rank, NewGenre, Runtime(Minutes), Votes, Revenue(Millions), and Metascore as the response variable. As the boosting model, in this case, was using a regression tree, no good results would once again be gained from using the NewGenre variable as the response as it is categorical. As with the regression tree model the standard deviation was calculated from the MSE and then compared to the value range of the response variable to determine if the boosting model was adequate at predicting the response variable. As the boosting models, in this case, used regression trees one would expect to adequately predict the same three response variable, which is the case. What is interesting though is that the only response variable that had a lower MSE and the standard deviation was Rating. This could indicate that the regression tree model was slightly overfitting on runtime and metascore. An interesting piece of data that can be gained from boosting is the relative influence of each predictor. In the case of Rating, the most influential predictor was metascore. This is not all that

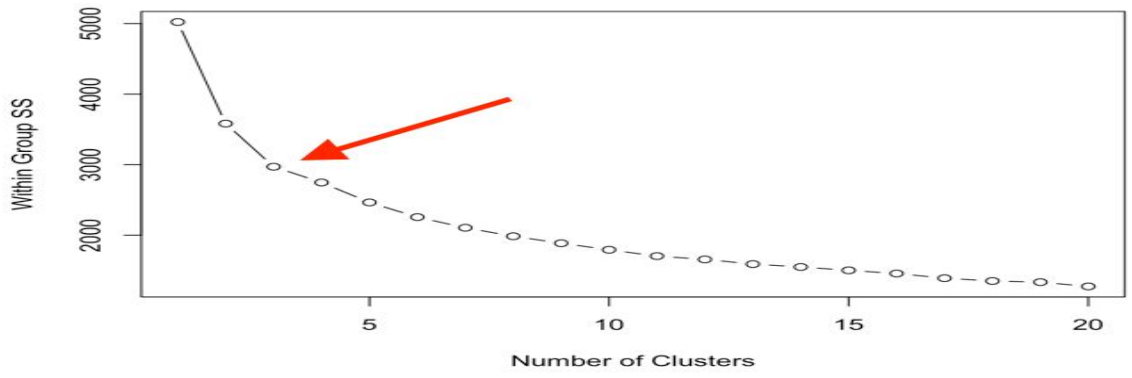
surprising as a metascore is typically just a weighted rating on a scale of 0-100. Surprisingly, however, for both runtime and metascore, the most influential predictor is NewGenre. For runtime this could be explained as the runtime for a film in a certain genre is fairly consistent. For metascore, this same argument could also be applied but more information might also be required.

K-Means Clustering:

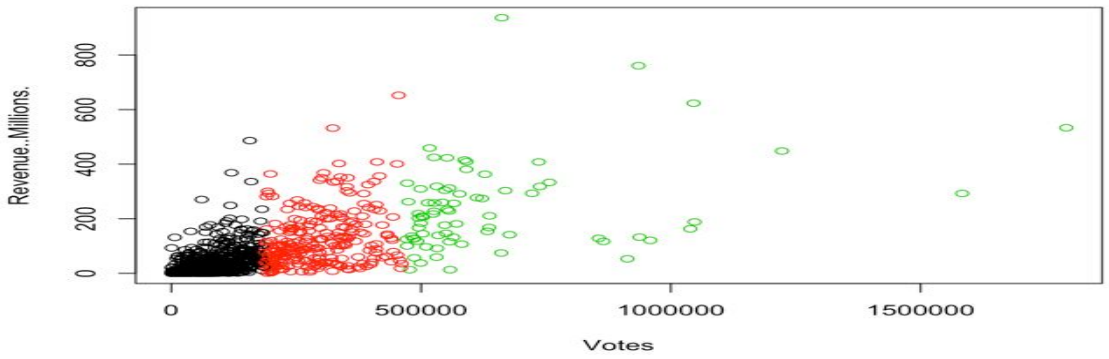
K-means clustering falls under unsupervised learning and is great for identifying unexpected segments or clusters in the dataset. K represents the number of groups, The algorithm is iterative and assigns data points to K groups or clusters based on a similar feature.

Analysis:

Two variables of Revenue and Votes are considered for K-means clustering. The goal is to find unidentified movie groups predicted by these two variables. To find the optimal number of K groups, within cluster sum of the square method (WCSS) has been used. The idea is minimizing the WCSS so that the distance between clusters can be maximized.



Using “Elbow method”, it can be said that the optimal number of clusters is 3. Therefore, the analysis is done using 3 groups.



The black cluster shows movies with lower votes and revenues. The film studios should try avoiding making these kinds of movies. On the other hand, the red cluster is a decent movie group consists of movies with slightly higher ratings and revenues. The green cluster is the most promising one and the film studios should focus on this while making their next box office hit. It is important to note the movie that generated the highest revenue is also in this cluster.

Item Based Collaborative Filtering:

The recommender engine uses item-based collaborative filtering. Collaborative filtering is a method that makes predictions of user interests by collecting preferences or taste information from many users. Item-based collaborative filtering is a type of collaborative filtering for recommender engines based on the similarity between items using user ratings.

Analysis:

Item based collaborative filtering is chosen instead of user based collaborative filtering since item contents are more stable than a living object like human. The items are movies in this case. A fictional user 0 is created for the purpose of analysis, someone who doesn't like horror movies, but enjoy family comedy and drama movies. A correlation matrix is used to find the movie similarities. In this case, the engine only considers movies that was rated by 10 users at least. The

parameter can be adjusted to get better recommendation results, but it is important to remember that the latest movies might not have 10 ratings. The system also predicts user ratings for a particular movie. The current top result for user 0 is Lord of War, which is also a drama biopic like the social network. The movies like Home Alone 2, The Little Rascals and Four Rooms are also similar to Home Alone. The other results also demonstrate movies related to the ones that the user highly rated and ignores horror movies. Thus, the engine is able to deliver decent recommendations successfully.

```
                User rated movies

Out[216]: title
          Grudge, The (2004)              1.0
          Home Alone (1990)               5.0
          Social Network, The (2010)      4.0
          Name: 0, dtype: float32
```

```
          sorting similar candidates by dropping the movies user already watched

Out[222]: Lord of War (2005)                                  4.701635
          Home Alone 2: Lost in New York (1992)               4.486614
          I Heart Huckabees (2004)                            4.008290
          RoboCop 2 (1990)                                    3.962715
          Four Rooms (1995)                                   3.920088
          Seabiscuit (2003)                                   3.807144
          'burbs, The (1989)                                  3.795971
          Little Rascals, The (1994)                          3.787380
          Sister Act 2: Back in the Habit (1993)              3.730924
          Mighty Morphin Power Rangers: The Movie (1995)      3.655491
```

Conclusion:

In conclusion, every model has its own importance and interpretation regarding the data. For instance, Lasso has been used to see if there is multicollinearity in the variables, if so either minimize the multicollinearity or solve the multicollinearity by suggesting only important variable according to the minimum CV MSE or CV within one standard error. Furthermore, the classification tree might have almost 56.1% error, but after pruning it provides specific genre according to the set of variables and given variables. As described above, Random Forest is good at predicting rating with the error of 33.6% training dataset and 35.5 % testing dataset.

Work Cited

Book:

James, Gareth, et al. *An Introduction to Statistical Learning with Applications in R*. Springer.

Lasso:

"Lasso Regression: Simple Definition." *Statistics How To*, 14 Oct. 2018, www.statisticshowto.datasciencecentral.com/lasso-regression/.

Rai, Bharatendra. *YouTube*, YouTube, 28 Mar. 2018, www.youtube.com/watch?v=_3xMSbIde2I.

Andrews, Jeff. RR and LASSO, https://canvas.ubc.ca/courses/22721/files/4192887?module_item_id=1071737

Multiple Linear Regression:

Shin, Lauren, and Lauren Shin. "Graphs and ML: Multiple Linear Regression." *Towards Data Science*, Towards Data Science, 30 July 2018, towardsdatascience.com/graphs-and-ml-multiple-linear-regression-c6920a1f2e70.

Andrews, Jeff. Multiple Linear Regression, https://canvas.ubc.ca/courses/22721/files/3734721?module_item_id=1005790 .

Rai, Bharatendra. *YouTube*, YouTube, 2 May 2015, www.youtube.com/watch?v=S-zKhFr91Tg.

Classification Tree:

Sanjeevi, Madhu, et al. “Chapter 4: Decision Trees Algorithms.” *Medium*, Deep Math Machine Learning.ai, 6 Oct. 2017, medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1.

Item Based Collaborative Filtering “Item-Item Collaborative Filtering.” *Wikipedia*, Wikimedia Foundation, 17 Feb. 2019, en.wikipedia.org/wiki/Item-item_collaborative_filtering.

Movielens Dataset: “MovieLens Latest Datasets.” *GroupLens*, 27 Sept. 2018, grouplens.org/datasets/movielens/latest/.

IMDB Dataset: “IMDB Data from 2006 to 2016.” *PromptCloud*, 26 June 2017, www.kaggle.com/PromptCloudHQ/imdb-data.