Upcoming Movie Recommendation System

Team Member

Hayoung Kim (hk2857)	Younhyuk Cho (yc3074)	Hyoungmook Song (hs2862)
Yunyi Zhang (yz2902)	Yang Fan (yf2361)	Haojiang Liu (hl2938)
Yang Fei (yf2372)	Yunlin Qin (yq2187)	Weitong Wang (ww2415)

Introduction And Objective

Audiences are often quite unsure which movie they should watch when they have some leisure time to step into the cinema. They are sometimes confused by the advertisements or marketing strategies and end up watching a movie that does not meet their expectation. An objective and rational recommendation system for upcoming movies based on the movie itself as well as audiences' own past preference is necessary in today's world.

Therefore, the objective of this project is to develop a system to recommend the upcoming movies based on the movie features along with audiences' previous preference.

Material And Method

1) Data Source

The data comses from Kaggle. The dataset includes different features of more than 5000 movies as well as their current IMDb scores. The movie features include Duration, Genres, Budget, Actor Facebook Likes, Aspect Ratio, Director Facebook Likes, etc.

2) Analytical Plans

The following supervised methods are used to construct a rating prediction model using current IMDb score as refered response variable:

- Multiple Linear Regression

- Ridge Regression - Lasso Regression

- Decision Tree

- Random Forest

- Support Vector Machine with Linear Kernel & Radial Kernel

Unsupervised clustering method is used to classify movies into six types. Movies in each type have similar features.

Analysis and Results

1) Data Processing

The following procedures are done for data processing.

- Delete data entries with missing values. Value of 0 is also considered missing because 0 is not a rational value in most of the variables.

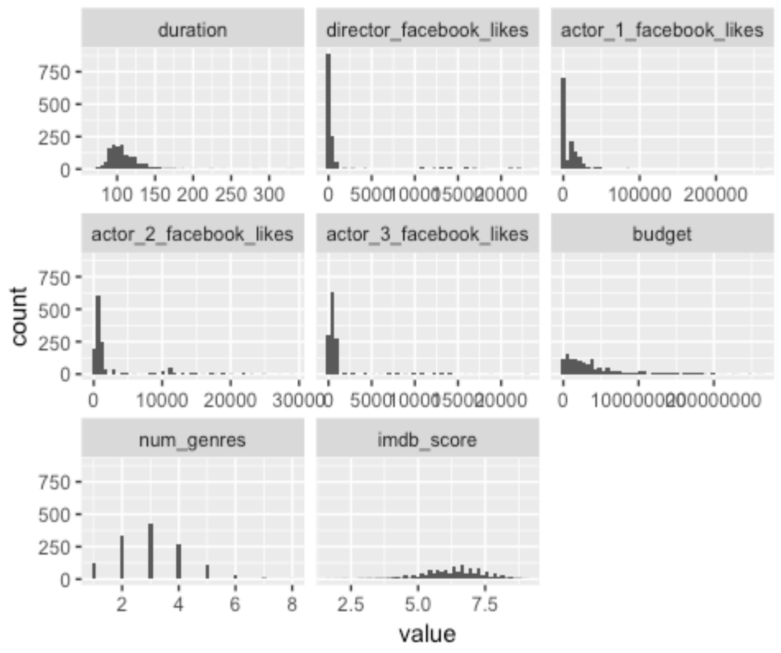
- Develop categorical variables like "content-rating" and "aspect-ratio" into cleansed factor variables ready for analysis.

- Transform genre variables from form of "Action/Comedy/Family" into "1/0" indicator variables for each of the 7 most popular genres.

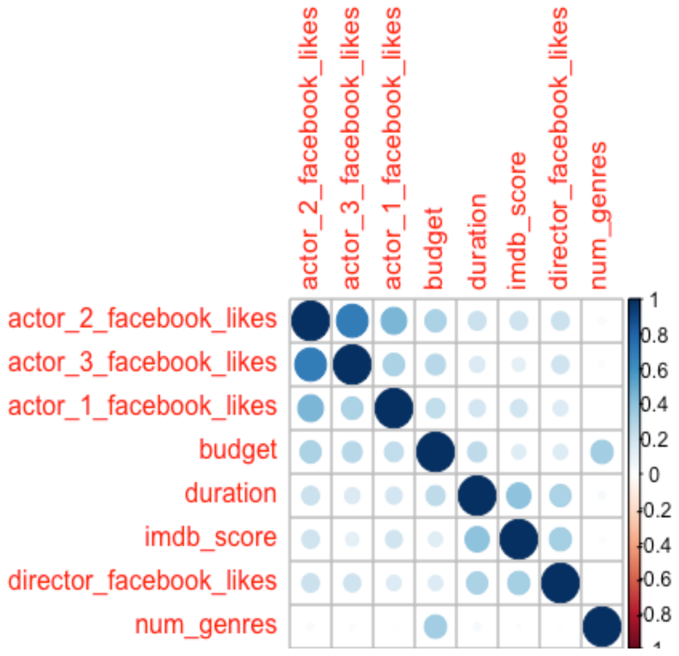
2) Exploratory Data Analysis for Numerical Variables

	Mean	**SD**	**Kurtosis**	**Skew ness**	**Min**	**Max**
Duration	108	21	22.95	3.15	69	330
Director Facebook Likes	1265	3962	12.37	3.65	2	23000
Actor_1 Facebook Likes	8399	12793	116.62	7.05	11	260000
Actor_2 Facebook Likes	2443	4537	6.79	2.66	9	29000
Actor_3_ Facebook Likes	1008	2356	27.58	4.98	3	23000
Budget	42277299	46303361	3.57	1.88	1100	263700000
Num_genre	3	1	0.11	0.41	1	8
Imdb_score	6	1	0.76	-0.54	2	9

Histogram



Correlation Matrix Plot



The position analysis and histogram of IMDb score show that it follows normal distribution to a great extent. Also, the correlations between the numerical predictors are relatively small except the correlation between actor_3_facebook_likes and actor_2_facebook_likes. This means that multiple regression is a valid method to build the predicting rating model.

3）Supervised Analysis

I. Multiple Linear Regression

```
#divide sample into training sets and test sets
sample_size<-floor(0.8*nrow(movie_dat2))
train_index<-sample(seq_len(nrow(movie_dat2)),size = sample_size)
train_data<-movie_dat2[train_index,]
test_data<-movie_dat2[-train_index,]

#fit linear regression on training sets and predict on test sets
ols_fit<-lm(imdb_score~.,data=train_data)
summary(ols_fit)
```

```
## Call:
## lm(formula = imdb_score ~ ., data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8953 -0.5286  0.0577  0.5835  2.5794
##
## Coefficients:
##                               Estimate      Std. Error t value
## (Intercept)              4.807427135631  0.195231883860  24.624
## duration                 0.009534646793  0.001566949556   6.085
## director_facebook_likes  0.000062995982  0.000007870710   8.004
## actor_3_facebook_likes  -0.000044422326  0.000017317415  -2.565
## actor_1_facebook_likes   0.000004830159  0.000002399324   2.013
## content_rating2          0.269072025464  0.102727440349   2.619
## content_rating3          0.710590487559  0.192929226239   3.683
## content_rating4          0.324501419993  0.069016828460   4.702
## content_rating5         -0.238631097859  0.922064937689  -0.259
## content_rating6          1.319149111900  0.567242353639   2.326
## content_rating7          0.484693955861  0.467187109758   1.037
## content_rating8          1.384560485312  0.518318673211   2.671
## content_rating9          1.320155906419  0.489699131989   2.696
## budget                   0.000000003090  0.000000000897   3.445
## actor_2_facebook_likes   0.000032326550  0.000009388838   3.443
## aspect_ratio2           -0.090505622248  0.064165302849  -1.411
## aspect_ratio3            0.130804109903  0.380743653113   0.344
## aspect_ratio4            0.594560432769  0.652521493737   0.911
## aspect_ratio5            0.008495510821  0.462595579171   0.018
## aspect_ratio6           -0.098907629999  0.282992928381  -0.350
## aspect_ratio7            0.465976287862  0.298519958102   1.561
## aspect_ratio8           -0.049036903716  0.305998023853  -0.160
## aspect_ratio9           -0.065698441137  0.463459292541  -0.142
## num_genres              -0.042596243464  0.036836490644  -1.156
## Action                  -0.220234718974  0.085757804511  -2.568
## Adventure                0.162699476221  0.099774835558   1.631
## Comedy                  -0.180773999127  0.075816182325  -2.384
## Crime                       0.156287578309  0.082897983182   1.885
## Drama                       0.500112753287  0.074645269747   6.700
## Romance                     0.080796083700  0.077714631233   1.040
## Thriller                   -0.167834602480  0.090995724299  -1.844
## Residual standard error: 0.9132 on 1015 degrees of freedom
## Multiple R-squared:  0.3328, Adjusted R-squared:  0.313
## F-statistic: 16.87 on 30 and 1015 DF,  p-value: < 0.0000000000000002
```

II. Stepwise Linear Regression

```
step<-stepAIC(ols_fit,direction = 'both',trace=0)
summary(step)

## 
## Call:
## lm(formula = imdb_score ~ duration + director_facebook_likes + 
##     actor_3_facebook_likes + actor_1_facebook_likes + content_rating 
+ budget + actor_2_facebook_likes + Action + Comedy + Crime + 
Drama + Thriller, data = train_data)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -5.0390 -0.5299  0.0814  0.5810  2.6160 
## 
## Coefficients:
##                               Estimate       Std. Error t value
## (Intercept)              4.6977792832729  0.1789291182973  26.255
## duration                 0.0100946161208  0.0015237509212   6.625
## director_facebook_likes  0.0000619066378  0.0000078277101   7.909
## actor_3_facebook_likes  -0.0000425514253  0.0000172355472  -2.469
## actor_1_facebook_likes   0.0000050204913  0.0000023892788   2.101
## content_rating2          0.2607464954608  0.0904001370431   2.884
## content_rating3          0.7212733484707  0.1723921887829   4.184
## content_rating4          0.3195160476485  0.0676731331636   4.721
## content_rating5         -0.2967545595257  0.9176652395855  -0.323
## content_rating6          1.3352661912926  0.5306364958870   2.516
## content_rating7          0.4583787206370  0.4612147279963   0.994
## content_rating8          1.4497706276877  0.4622939875611   3.136
## content_rating9          1.2675832084130  0.4639935034360   2.732
## budget                   0.0000000032936  0.0000000008143   4.045
## actor_2_facebook_likes   0.0000322694668  0.0000093543508   3.450
## Action                  -0.2231645802561  0.0791333432715  -2.820
## Comedy                  -0.2346239410316  0.0698447925076  -3.359
## Crime                    0.1231981659024  0.0796005632348   1.548
## Drama                    0.4583720656526  0.0673886950196   6.802
## Thriller                -0.2317648634271  0.0790229028278  -2.933
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
```

```
## Residual standard error: 0.9125 on 1026 degrees of freedom
## Multiple R-squared:  0.3265, Adjusted R-squared:  0.3141
## F-statistic: 26.18 on 19 and 1026 DF,  p-value: < 0.0000000000000002

ols_step_pred<-predict(step,test_data[,-8])
```

The stepwise regression using AIC as standard returns a model with predictors: duration, director facebook likes, actor_3 facebook likes, actor_2 facebook likes, actor_1 facebook likes , content rating, budget and five genre indicators including Action, Comedy, Crime, Drama and Thriller.

III. Ridge & Lasso Regression

```
train_data$content_rating<-as.numeric(train_data$content_rating)
test_data$content_rating<-as.numeric(test_data$content_rating)
ridge_cv_fit<-cv.glmnet(x=as.matrix(train_data[,-
c(8,9)]),y=train_data[,8],alpha=0)
ridge_fit<-glmnet(x=as.matrix(train_data[,-8]),y=train_data[,8],alpha=0)

op_ridge<-which(ridge_cv_fit$lambda==ridge_cv_fit$lambda.min)
ridge_fit$beta[,op_ridge]

## duration            director_facebook_likes  actor_3_facebook_likes
## 0.009699145530368       0.000059390464639      -0.000033967701539
## actor_1_facebook_likes     content_rating              budget
## 0.000004902402505       0.114089007236396       0.000000002749191
## actor_2_facebook_likes        num_genres                   Action
## 0.000026345184883       0.007281398243364      -0.263889161637586
## Adventure                    Comedy                    Crime
## 0.187700948507112      -0.226978838246522       0.116628923302687
## Drama                          Romance                  Thriller
## 0.418093393387702       0.045292533810239      -0.234499019885354
```

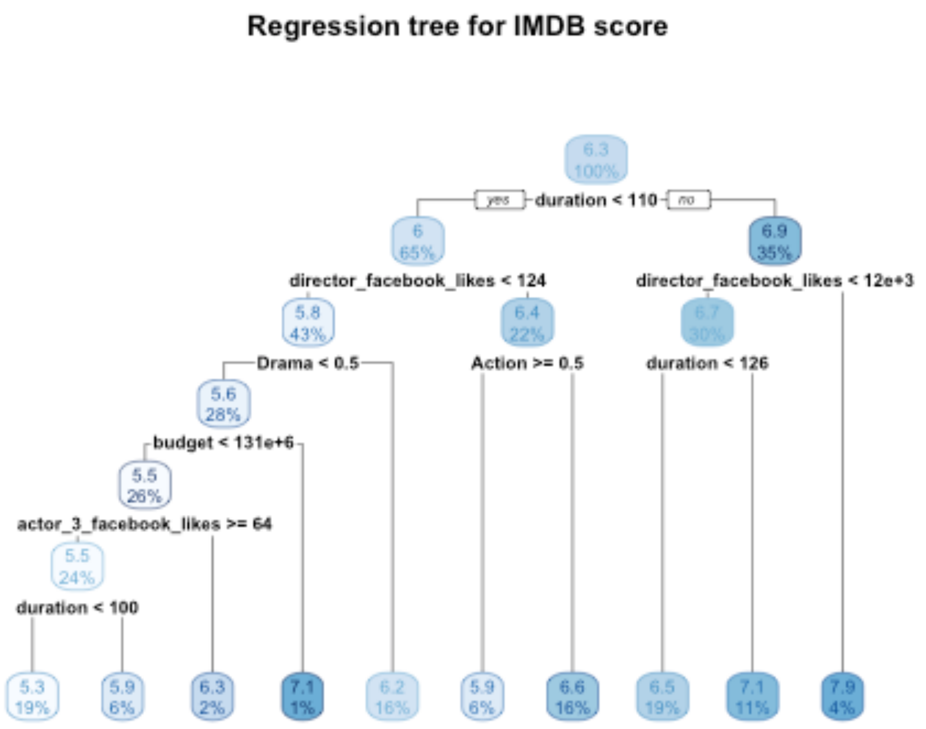
```
r_pred<-predict(ridge_cv_fit,as.matrix(test_data[,-c(8,9)]))

lasso_cv_fit<-cv.glmnet(x=as.matrix(train_data[,-c(8,9)]),y=train_data[,
8],alpha=1)
lasso_fit<-glmnet(x=as.matrix(train_data[,-c(8,9)]),y=train_data[,8],al
pha=1)
op_lasso<-which(lasso_cv_fit$lambda==lasso_cv_fit$lambda.min)
lasso_fit$beta[,op_lasso]

##  duration      director_facebook_likes       actor_3_facebook_likes
##  0.009824465450748       0.000061964018736      -0.000040723120630
##  actor_1_facebook_likes     content_rating              budget
##  0.000004686160652       0.121850106087329       0.000000003054157
##  actor_2_facebook_likes     num_genres                   Action
##   0.000029016537808       0.002589086925356      -0.278071162779997
##    Adventure                    Comedy                   Crime
##    0.204204708763764      -0.226340970781899       0.126473451225435
##    Drama                        Romance                     Thriller
##    0.443896603699231       0.049572212088111      -0.243828153634657
```

For Ridge&Lasso Regression, categorical variables cannot be used. Aspect ratio is removed according to the result of multiple linear regression. Content rating is transformed to numerical variable to facilitate ridge and lasso regression analysis.

Lasso Regression returns a model with predictors: duration, director_facebook_likes, actor_3_facebook_likes, actor_2_facebook_likes, actor_3_facebook_likes, content_rating, budget, number of genres, and all of the seven genre indicators.

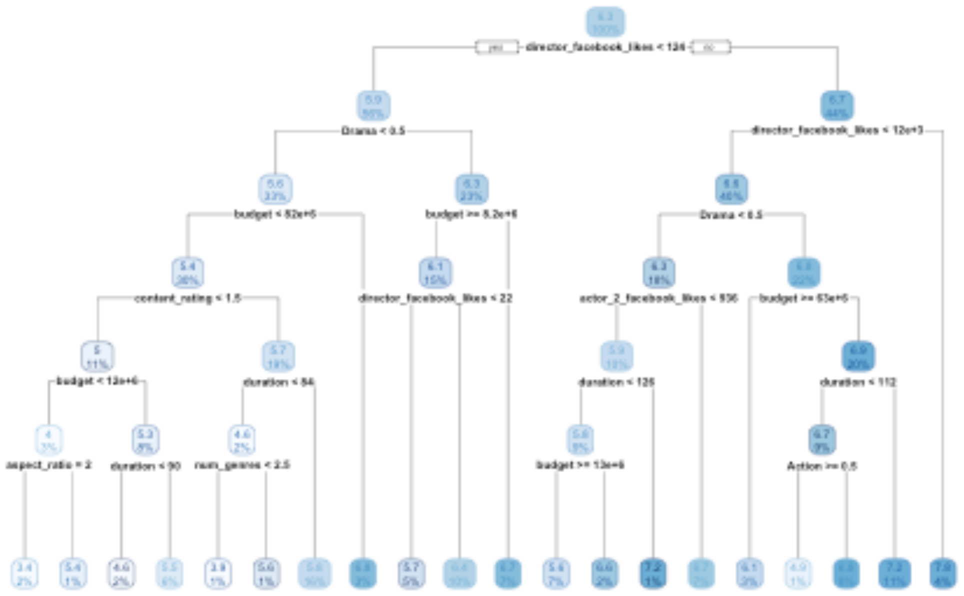
IV. Decision Tree

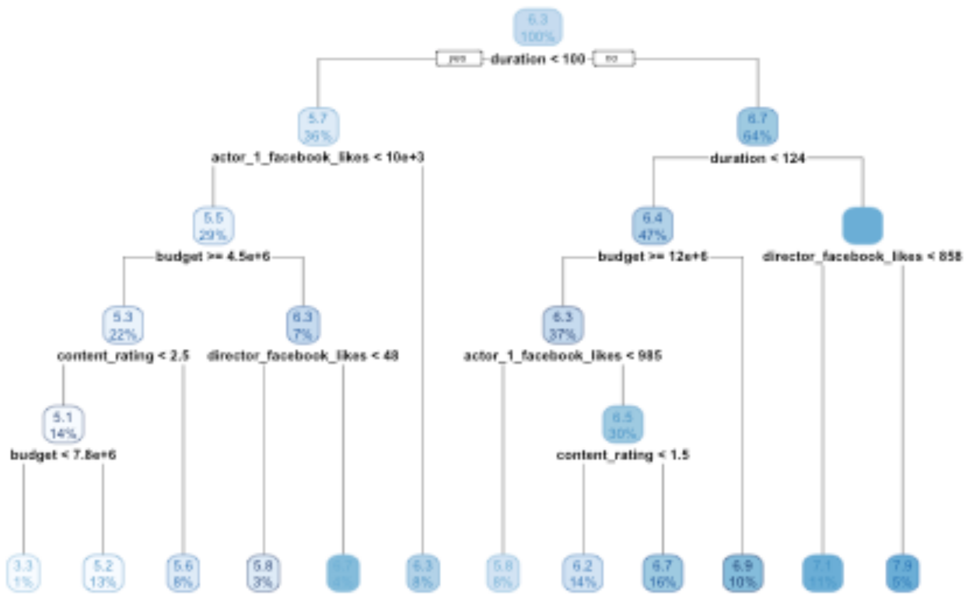
```
########### 4. Fit decision trees on data
tree_movie<-rpart(imdb_score~.,data=train_data)
rpart.plot(tree_movie,main="Regression tree for IMDB
score",col=blues9[5:9])
```



V. Bagging Decision Tree

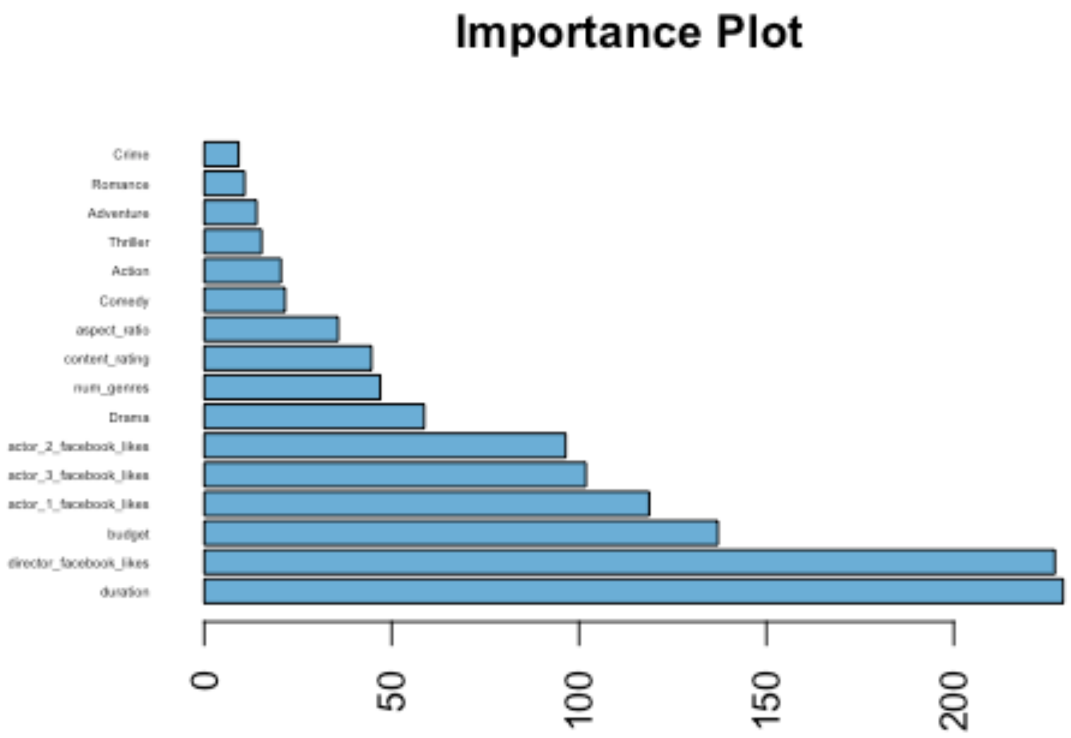
```r
K<-500
out.vals<-mat.or.vec(262,K)
out.list<-list(K)
base.fit<-rpart(imdb_score~.,data=train_data)
out.vals.base<-predict(base.fit,test_data)

set.seed(1)
for (i in 1:K){
  inds<-sample(1:1046,1046,replace=TRUE)
  df.temp<-train_data[inds,]
  fit.temp<-rpart(imdb_score~.,data=df.temp)
  out.list[[i]]<-fit.temp
  out.vals[,i]<-predict(fit.temp,test_data)
}
#calculate the mean MSE of all bagging trees
bag.mse.vec<-rep(0,262)
for (i in 1:262){
  bag.mse.vec[i]<-mean((out.vals[,i]-test_data$imdb_score)^2)
}
bag_mse<-mean(bag.mse.vec)
rpart.plot(out.list[[6]],col=blues9[5:9])
```





VI. Random Forest

```
#use random forest on our data and calculate MSE
rf_movie<-randomForest(imdb_score~.,data=train_data)
y_pred_rf<-predict(rf_movie,test_data)
mse_rf<-mean((y_pred_rf-test_data$imdb_score)^2)

#show the importance of variables using random forest
importance(rf_movie)

##                          IncNodePurity
## duration                    229.171774
## director_facebook_likes     226.970069
## actor_3_facebook_likes      101.820243
## actor_1_facebook_likes      118.727533
## content_rating               44.631328
## budget                      137.057467
## actor_2_facebook_likes       96.426970
## aspect_ratio                 35.731808
## num_genres                   46.954666
## Action                       20.452542
## Adventure                    13.964077
## Comedy                       21.574172
## Crime                                 9.155173
## Drama                                58.690743
## Romance                              10.758747
## Thriller                             15.279852

imp<-importance(rf_movie)
imp_sort<-sort(imp,decreasing=TRUE,index.return=TRUE)

#draw a barplot to show the importance of the variables
par(las=2)
barplot(imp_sort$x,main="Importance
Plot",horiz=TRUE,names.arg=rownames(imp)[imp_sort$ix],cex.names=0.35,co
l=blues9[5])
```



VII. Support Vector Machine (Linear & Radial Kernel)

```
fit_svm_linear <-svm(imdb_score~.,data=train_data,kernel="linear")
fit_svm_radial <-svm(imdb_score~.,data=train_data,kernel="radial")
y_pred_svm_l<-predict(fit_svm_linear,newdata=test_data)
y_pred_svm_r<-predict(fit_svm_radial,newdata=test_data)
mse_svm_l<-mean(y_pred_svm_l-test_data$imdb_score)
mse_svm_r<-mean(y_pred_svm_r-test_data$imdb_score)
```

VIII. Supervised Methods Result

Model	Test Set MSE	Prediction Accuracy	Interpretability
Multiple Linear Regression	0.944	Poor	Good
Stepwise Linear Regression	0.926	Poor	Good
Ridge Regression	0.963	Poor	Good
Lasso	0.922	Fair	Good
Decision Tree	0.901	Fair	Good
Bagging	0.995	Fair	Fair
Random Forest	0.747	Good	Poor
Support Vector Machine (Linear)	0.931	Poor	Poor
Support Vector Machine (Radial)	0.827	Good	Poor

Both Stepwise and Lasso Regression perform variable selection. Lasso performs better in this case. The model returned by Random Forest gives the best prediction capability but it is a hard-to-interpret model. The model returned by Decision Tree is recommended if interpretability is considered important.

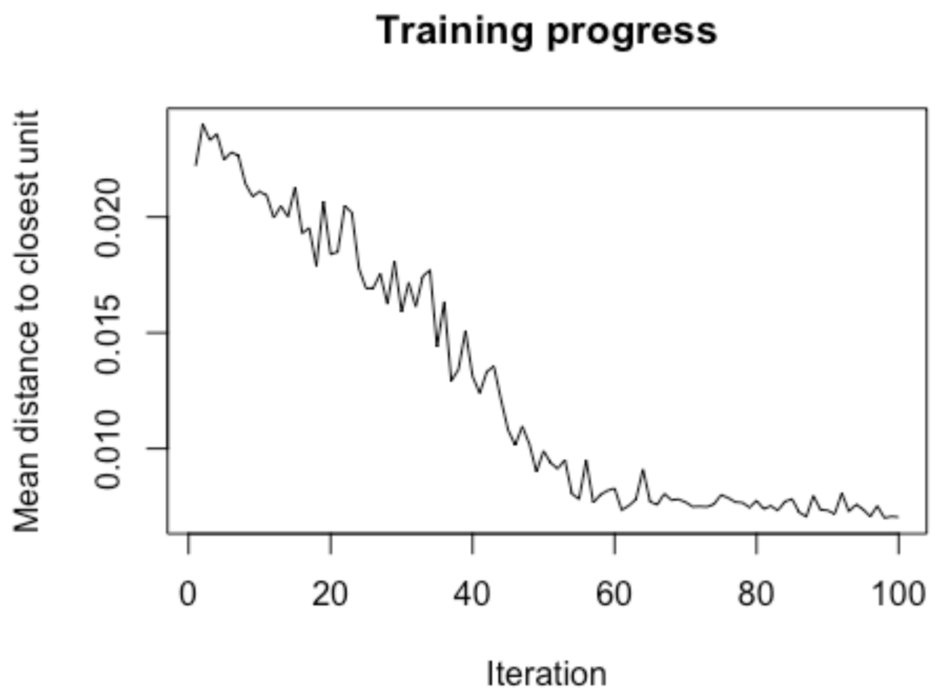
4) Unsupervised Clustering

In this part, hierarchical clustering along with Self Organizing Maps as visualization tool is used to classify movies into 6 types. Each type has specific characteristics.

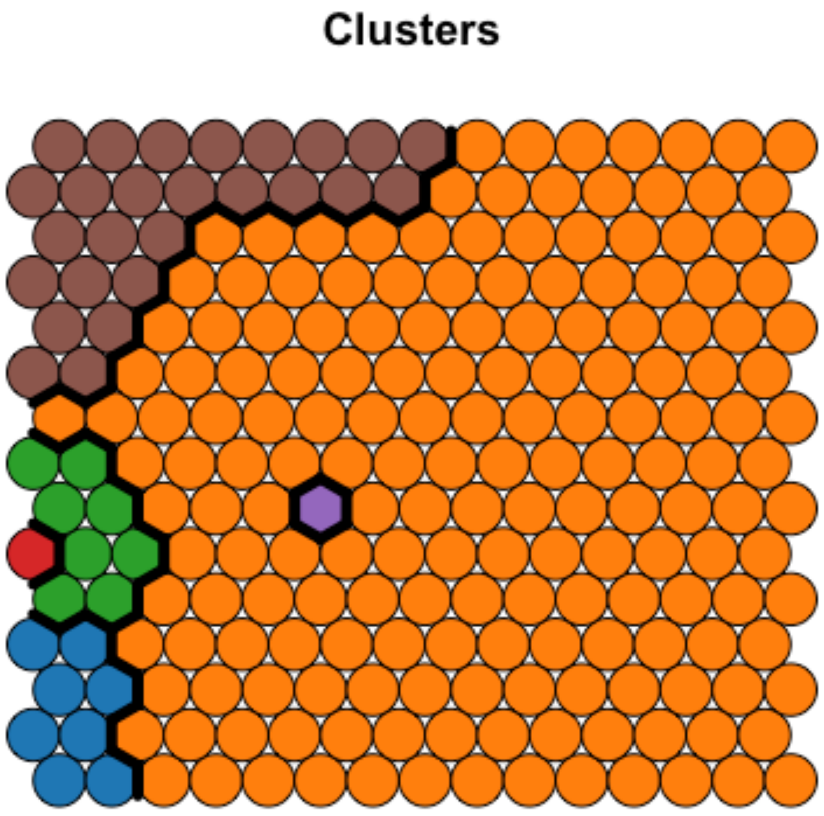
```r
#text(x = midpts, y = barplot.data, label = barplot.data, pos = 1, cex
#use the subset of data and apply SOM
set.seed(10)
movie.data$color<-as.numeric(movie.data$color)
movie.data$color<-movie.data$color-2
content_rating<-as.numeric(movie_dat$content_rating)
target.movie.data.som
=cbind(movie.data[,c(1,4,5,8,14,23,29)],content_rating,aspect_ratio)
training.data.som = scale(target.movie.data.som)
som.movie = som(training.data.som, grid = somgrid(15, 15, "hexagonal"),
rlen=100, alpha=c(0.05, 0.01))
summary(som.movie)
```

```
## som map of size 15x15 with a hexagonal topology.
## Mean distance to the closest unit in the map: 0.6345163
```

```r
coolBlueHotRed <- function(n, alpha = 1) {
        rainbow(n, end=4/6, alpha=alpha)[n:1]}
#plot the iteration of the algrithms
par(mfrow = c(1,1))
plot(som.movie, type="changes")
```



```r
#use hierarchical clustering to cluster the codebook vectors
par(mfrow = c(1,1))
pretty_palette = c("#1f77b4", '#ff7f0e', '#2ca02c', '#d62728',
'#9467bd', '#8c564b', '#e377c2')
som.cluster = cutree(hclust(dist(som.movie$codes)), 6)
plot(som.movie, type="mapping", bgcol = pretty_palette[som.cluster],
main = "Clusters")
add.cluster.boundaries(som.movie, som.cluster)
```



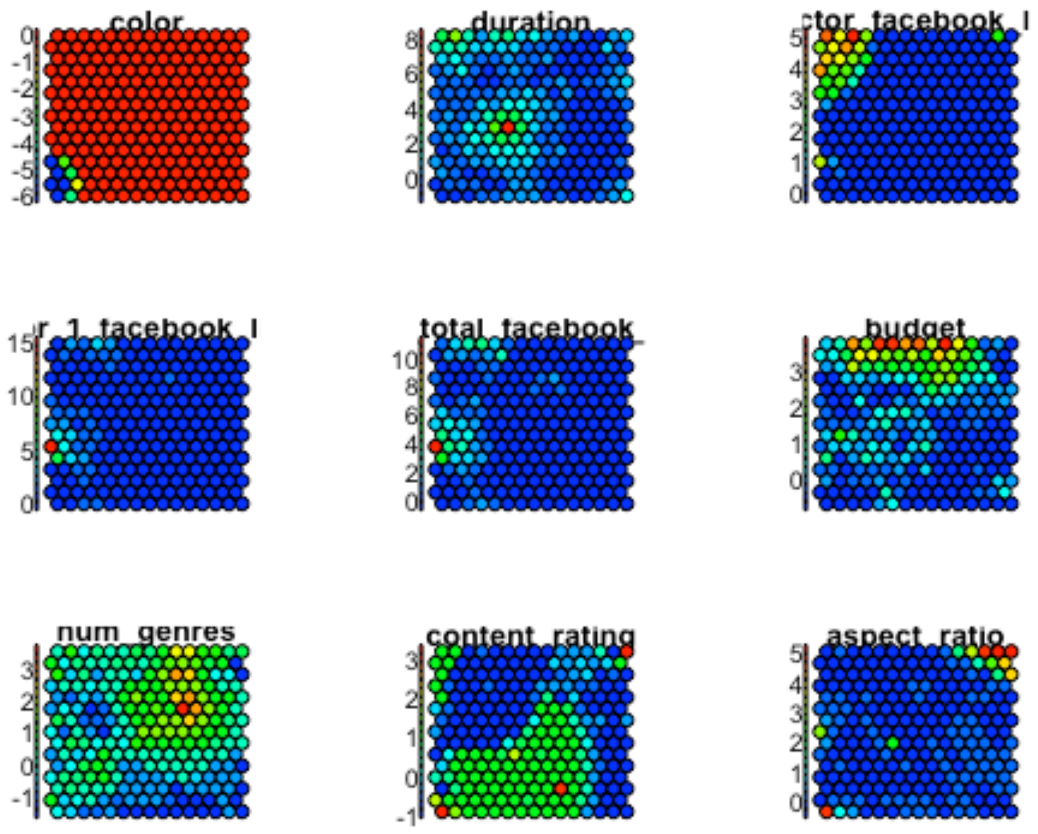
In this graph, each circle contains certain number of movies in the sample. After hierarchical clustering is run, all of the movies are classify into six types, which are represented with six different colors in the graph.

```r
movie.data$movie_title[(som.movie$unit.classif == 96)]

## [1] Gods and Generals   Heaven's Gate       Blood In, Blood Out
## [4] Gone with the Wind
```

For example, the purple color circle contains the following movies: “Gods and Generals”, “Heaven’s Gate”, “Blood In, Blood Out”, “Gone with the Wind”.

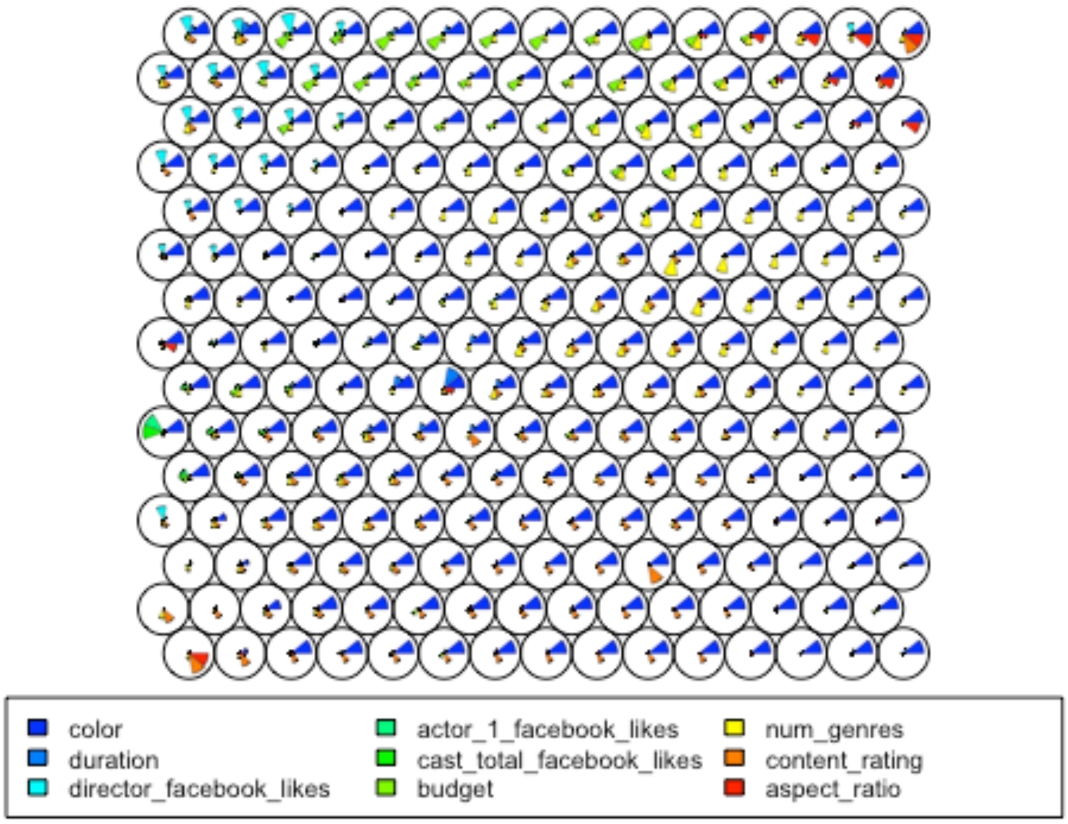
The followings are the heat maps of the 9 predictors. The red color in the circle means that the movies in the circle have the highest value in the corresponding predictor. For example, the movies in the upper left circles have more director Facebook likes.

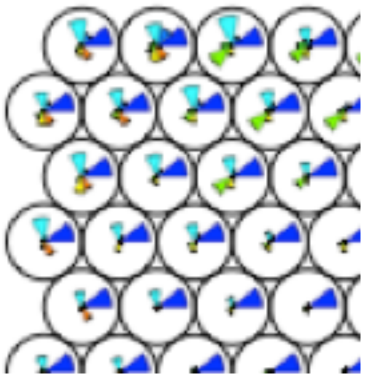
```r
#show different features in 9 plots using SOM map
pretty_palette = c("#1f77b4", '#ff7f0e', '#2ca02c', '#d62728',
'#9467bd', '#8c564b', '#e377c2')
par(mfrow = c(3,3))
for (i in 1:9){plot(som.movie, type = "property", property =
som.movie$codes[,i], main=colnames(som.movie$data)[i],
palette.name=coolBlueHotRed)}
```



```r
#show all the features in one plot using SOM map
par(mfrow = c(1,1))
plot(som.movie, type="codes",palette.name = coolBlueHotRed)
```

The following graph puts all of the heat maps together:





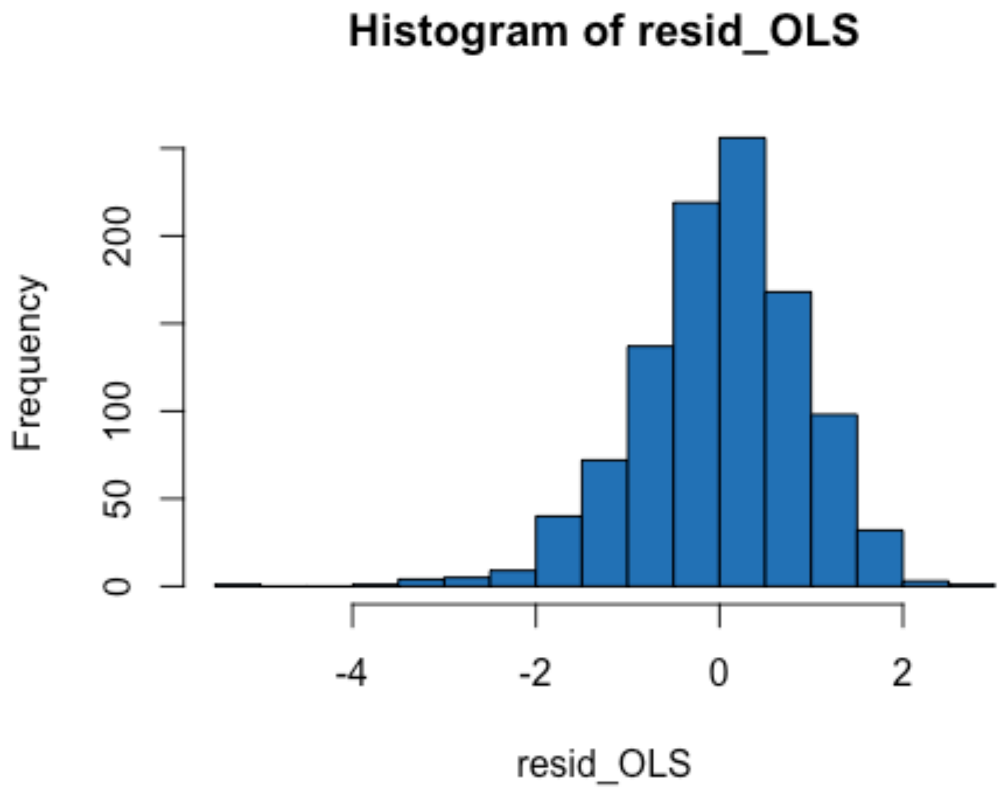
In each circle, 9 fan shapes correspond to the 9 predictors. The larger the fan, the higher the value of the corresponding variable is.

As mentioned before, the movies in the upper left circles has higher director_facebook_likes and they are all color movies.

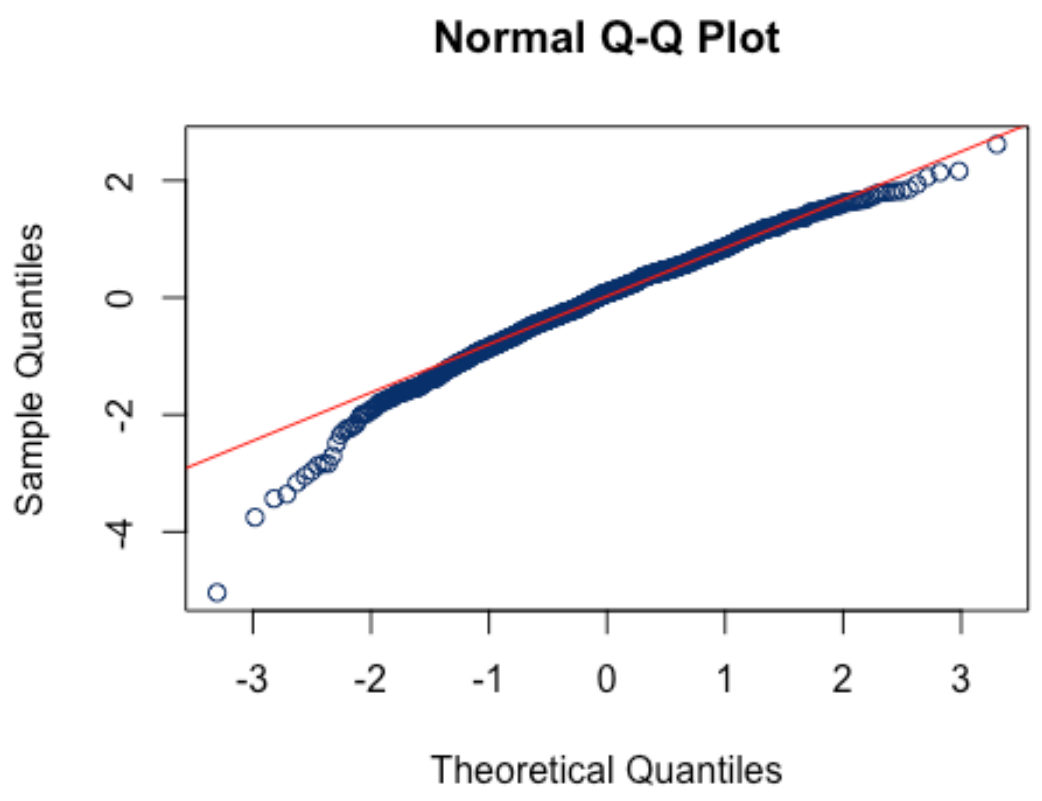
Assumption Check For Linear Model

```
vif(step)

##                              GVIF Df GVIF^(1/(2*Df))
## duration                 1.393962  1        1.180662
## director_facebook_likes 1.135279  1        1.065495
## actor_3_facebook_likes  1.985497  1        1.409077
## actor_1_facebook_likes  1.279674  1        1.131227
## content_rating          1.544251  8        1.027531
## budget                  1.760475  1        1.326829
## actor_2_facebook_likes  2.281512  1        1.510468
## Action                  1.454111  1        1.205865
## Comedy                  1.511921  1        1.229602
## Crime                   1.249643  1        1.117874
## Drama                   1.422969  1        1.192883
## Thriller                1.534324  1        1.238678

resid_OLS<-resid(step)
hist(resid_OLS,col=blues9[7])
```

The VIFs are all relatively small. Therefore, there exists little multicollinearity.



```
qqnorm(resid_OLS,col=blues9[9])
qqline(resid_OLS,col="red")
```



```
shapiro.test(resid_OLS)

##
##  Shapiro-Wilk normality test
##
## data:  resid_OLS
## W = 0.97998, p-value = 0.0000000000812
```

The histogram and qqnorm plot of the residuals show that the residuals are slightly left skewed. The Shapiro Test also shows that the normality assumption in the linear model is violated with a very small p-value.

```
ncvTest(step)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 10.83869    Df = 1     p = 0.0009940115
```

The non-constant variance assumption is also violated with a small p-value.

```
durbinWatsonTest(step)

##  lag Autocorrelation D-W Statistic p-value
##    1      0.02797887      1.939384    0.29
##  Alternative hypothesis: rho != 0
```

The result of Durbin Watson Test shows that there is little autocorrelation in the residuals.

Conclusion

1) Summary

With the supervised model building and unsupervised clustering, an upcoming movie recommendation system is constructed. Upcoming movies that are in the same type as the user's past preference are listed. Then, based on the rating model, the one with the highest rating among the listed movies is recommended to the user.

For the rating model, random forest algorithm is recommended to build an accuracy-oriented model.

2) Limitation

I. The supervised model uses IMDb score as a reference respond variable. However, the IMDb score may not be an absolute objective rating that represents the opinions of the whole population. Therefore, the predicted rating may not be fully convincible.

II. An audience may not always want to watch the same type of movies that he or she likes before. From this aspect, the clustering model may not be useful enough to recommend the right movies.

III. The dataset is relatively small. More samples may be needed for a better model building.

References

1) Please refer to https://www.kaggle.com/deepmatrix/imdb-5000-movie-dataset for the detailed description of the dataset.

2)Please refer to https://cran.r-project.org/web/packages/som/som.pdf for the detailed description of the SOM-Clustering algorithm and R Package used in the analysis.

Appendix

Please refer to https://github.com/yunyizh/adaproject for the coding file and a copy of the report.