# An Empirical Analysis of IMDB Movie Dataset

Yue Shi
Dayu Zhong
Yifei Sun
Zhangzhi Cao

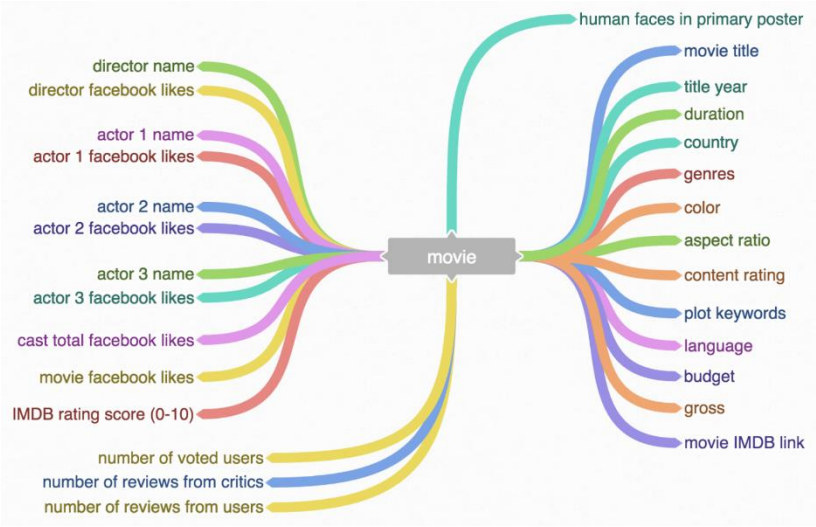
3/27/17

## Content Overview

## Summary

We obtained 28 variables in the movie dataset from 5043 movies and 4906 posters, spanning across 100 years in 66 countries. For example, there are 2399 unique director names, and thousands of actors/actresses.

Our goal is to predict the IBDB Score of a movie based on the following attributes:



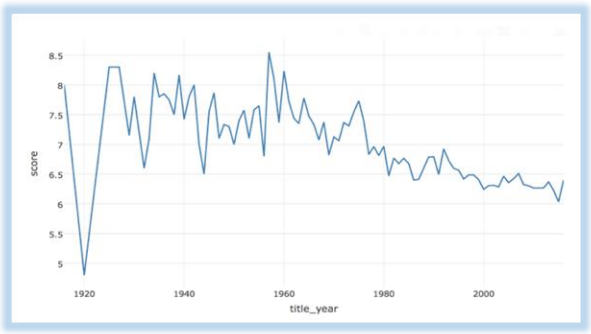

In the report, we first considered the linear regression model, where we conducted both variable and model selection using subset selection and stepwise method. We also utilized ridge and lasso regression to include all the variables, and we found that lasso regression performs well in our case as it forces some of the coefficients to be 0 and therefore mitigated the problem of multicollinearity and overfitting.

We then considered the general additive model in additional to the multiple linear regression. For each variable, we performed curve fitting and compared model variance of different non-linear functions to select the best model while avoiding the suspicion of overfitting. Surprisingly, we found that the general additive model has some improvement in training error, but fails to improve on our test data, with a slightly higher test error than the saturated linear model.

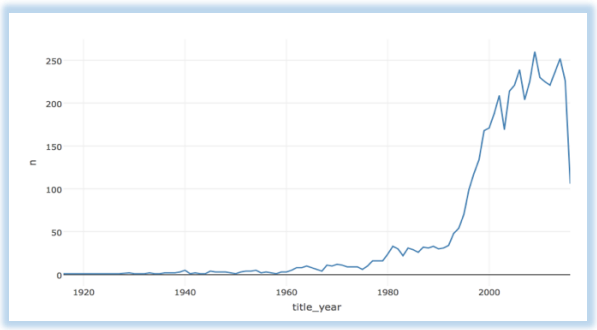
Lastly, we considered non-parametric approaches including principle component analyses and tree methods. With a few components, the principle regression analysis performs the worst among all the models. Partial least squares, on the other hand, significant improves over the principle regression analysis and has the training and test error comparable to the linear models.

For the tree methods, we considered regression tree, tree pruning, bagging, random forests and boosting. To our surprise, the tree methods have in general yielded more than 30% improvement over our previous models in terms of mean square of errors (MSE). This non-parametric approach beats all the regressions we conducted previously.

# Exploratory Data Analysis



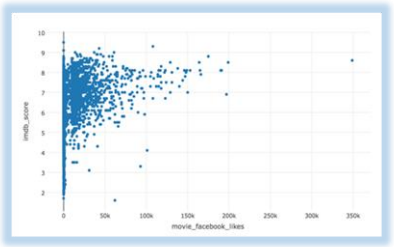

Trend of Average IMDB Rating Score by Year



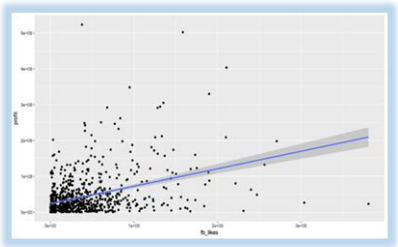

Trend of Movie Reviews Numbers by Year

The overall trend of average IMDB score appears to be declining. The highest average score was achieved in the year 1957, with the lowest being achieved in 1920.

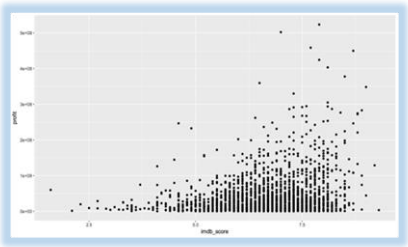
In addition, as time went on, the number of movie reviews increased dramatically, especially after 1990s. From the plot above, we could see that the trend seems to be exponential. The highest number of movie reviews were in the year 2009, with the number being 260.



Facebook Likes VS IMDB Score



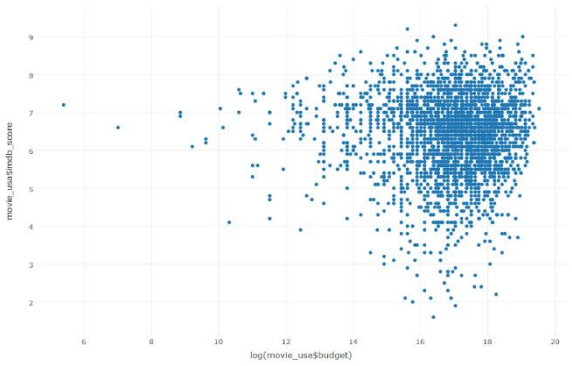
Facebook Likes VS Movie Profit

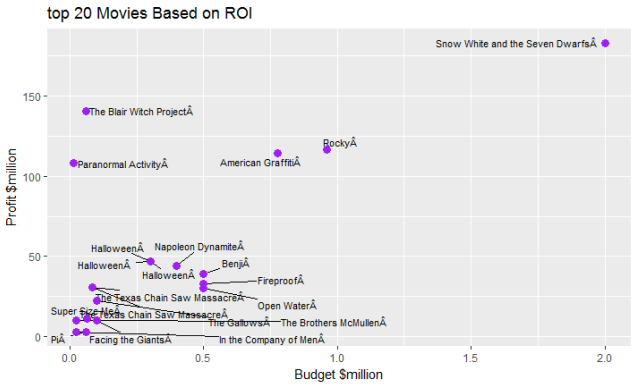


IMDB Score VS Movie Profit

Since IMDB Score and profit made by the movie are two important variables that we care about most, we further explore the relationship among movie's Facebook likes number, profit, and IMDB Score. To get the total number of Facebook likes, we add the two variables cast_total_facebook_likes and movie_facebook_likes together.

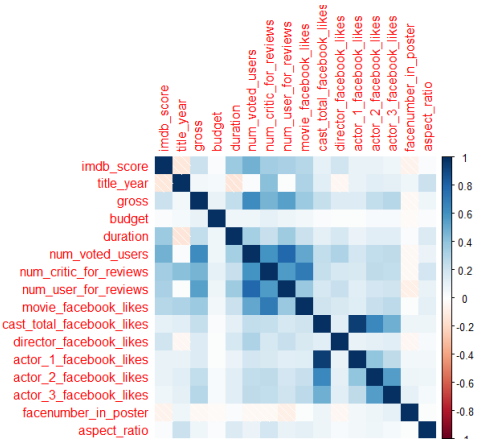
There is a clear positive correlation between budget and ratings. It's safe to say that for the most part, IMDB users enjoy big budget movies.




A plot of the top 20 films based on ROI = (Net Profit / Budget) x 100, the ROI based plot is interesting as we start to see films like The Blair Witch Project and Paranormal Activity which had extremely low budgets but were extremely profitable. Horror films really seem to shine here.

For each movie, there are a few words to describe the movie plot. We make a word cloud plot to explore the text variable plot. Most movies are about love, murder, friend and death.



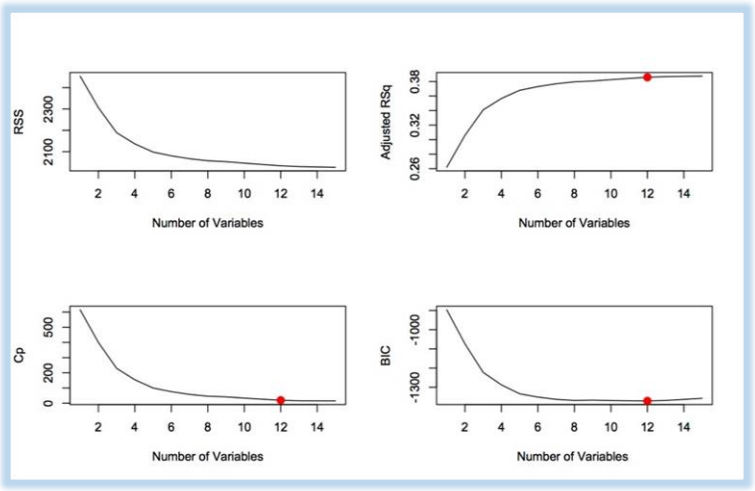


## Best Subset Selection

The correlation matrix plot show thats multicollinearity exists in the 15 continuous variables. When fitting a multiple linear regression model to predict movie rating, we need to further remove some variables to reduce multicollinearity. Thus, here we discuss some ways in which the simple linear model can be improved, by replacing plain least squares fitting with some alternative fitting procedures.

We first choose to do the best subset selection using adjusted $R^2$, $C_P$ and BIC as criteria. This approach involves identifying a subset of the predictors that we believe to be related to the response.

The figure below shows that, as expected, these quantities improve as the number of variables increases; however, from the 12-variable model on, there is little improvement as a result of including additional predictors. In this case, three variables are removed. They are director_facebook_likes, actor3_facebook_likes, and aspect_ratio. This result agrees with the p-value from our initial simple linear regression, as it shows that these three variables are not significant in the model.



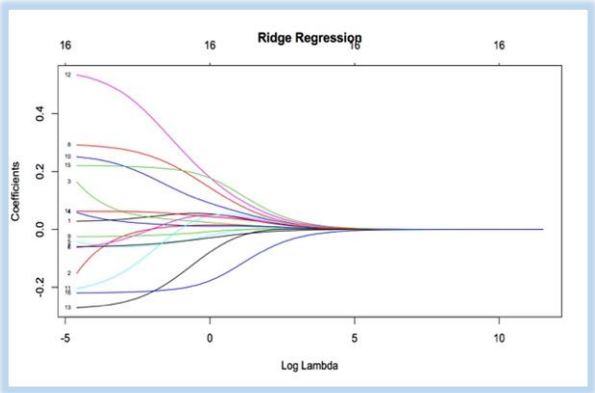

## Stepwise Selection

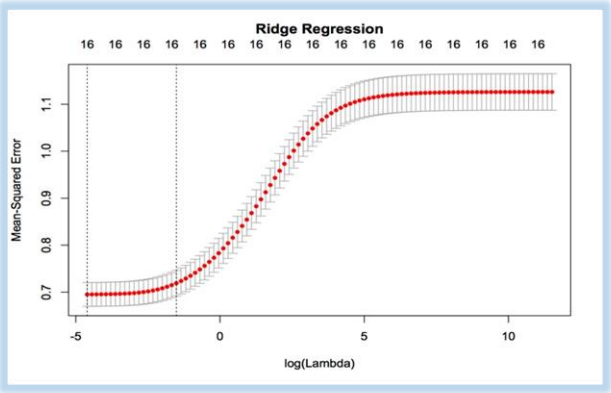
We also use forward stepwise selection with AIC as criteria to do the variable selection, which is a computationally efficient alternative to best subset selection. In this model, four predictor variables are removed, which is slightly different from the result of best subset selection. Besides the two variables (director_facebook_likes, aspect_ratio) in common with best subset selection, cast_total_facebook_likes and actor2_facebook_likes are also removed. This is reasonable since the variable actor3_facebook_likes is highly correlated with cast_total_facebook_likes and actor2_facebook_likes.

The two models we describe above all belong to the subset selection methods, which involve using least squares to fit a linear model that contains a subset of the predictors. As an alternative, we can fit a model containing all predictors using a technique that constrains or regularizes the coefficient estimates, or equivalently, that shrinks the coefficient estimates towards zero. It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance. The two best-known techniques for shrinking the regression coefficients towards zero are ridge regression and the lasso.

## Ridge Regression

Ridge Regression is a regularization method that tries to avoid overfitting, penalizing large coefficients through the L2 Norm. Its advantage over least squares is rooted in the bias-variance trade-off. As $\lambda$ increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias.



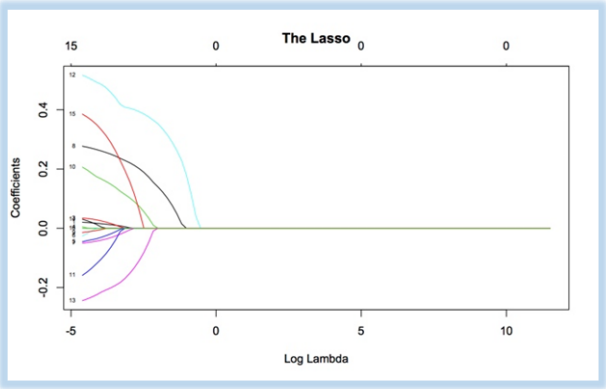

In the left-hand panel, the ridge regression coefficient estimates for the dataset are displayed. Each curve corresponds to the ridge regression coefficient estimate for one of the 15 variables, plotted as a function of log λ. At the extreme left-hand side of the plot, λ is essentially zero, and so the corresponding ridge coefficient estimates are the same as the usual least squares estimates. But as λ increases, the ridge coefficient estimates shrink towards zero. When λ is extremely large, then all of the ridge coefficient estimates are basically zero; this corresponds to the null model that contains no predictors.

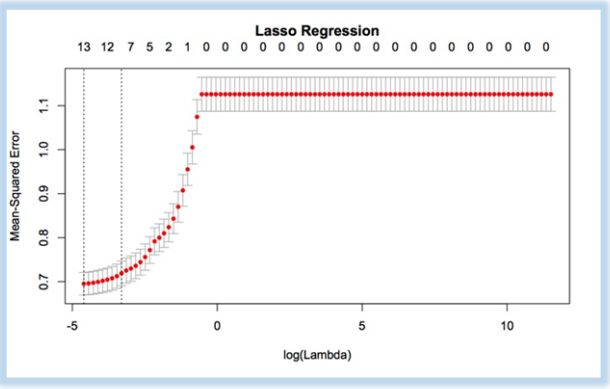
The right-hand panel shows the train MSE corresponding to different values of log λ. Instead of arbitrarily choosing the tuning parameter λ, we use ten-fold cross-validation to choose the best λ. To create training and testing datasets, we decide to use a 70-30 split with approximately 70% of our data in the training set and 30% of our data in the test set. We find that the value of λ that results in the smallest cross-validation error is 0.1149757. The test MSE associated with this value of λ is 0.6764366.

However, ridge regression does have one obvious disadvantage. Unlike best subset and forward stepwise selection, which will generally select models that involve just a subset of the variables, ridge regression will include all predictors in the final model. This may not be a problem for prediction accuracy, but it can create a challenge in model interpretation in settings in which the number of variables is quite large. Thus, we ask whether the lasso can yield either a more accurate or a more interpretable model than ridge regression.

## Lasso Regression

Lasso is also a regularization method that tries to avoid overfitting penalizing large coefficients, but it uses the L1 Norm. As with ridge regression, the lasso shrinks the coefficient estimates towards zero. However, in the case of the lasso, the L1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when λ is sufficiently large. Hence, much like best subset selection, the lasso performs variable selection. As a result, models generated from the lasso are generally much easier to interpret than those produced by ridge regression.




We can see from the coefficient plot that depending on the choice of tuning parameter, some of the coefficients will be exactly equal to zero. When $\lambda = 0$, then the lasso simply gives the least squares fit, and when $\lambda$ becomes sufficiently large, the lasso gives the null model in which all coefficient estimates equal zero. Hence, depending on the value of $\lambda$, the lasso can produce a model involving any number of variables.

As with ridge regression, selecting a good value of $\lambda$ for the lasso is very critical. We also perform ten-fold cross-validation and compute the associated test error. We find that the value of $\lambda$ that results in the smallest cross-validation error is 0.0191791. The test MSE associated with this value of $\lambda$ is 0.6792812. This is substantially lower than the test set MSE of the null model and of least squares, and very similar to the test MSE of ridge regression with $\lambda$ chosen by cross-validation. However, the lasso has a substantial advantage over ridge regression in that the coefficient of the variable cast_total_facebook_likes become zero, thus performing a selection of attributes with the regularization.

To do the model selection, we compare the test MSE of each model. We also refit the above four models on the full data set and computed the MSE of each model on all the data. The table below shows the summary.

| | Best Subset | Stepwise | Ridge | Lasso |
|---|---|---|---|---|
| Test MSE | 0.6748232 | 0.6763817 | 0.6764366 | 0.6792812 |
| MSE (full dataset) | 0.6923257 | 0.6956305 | 0.6832386 | 0.6831095 |

Looking at the the test MSE, we did not find a significant difference among these four models. As for the MSE on the full data set, the lasso performed better than the other three models. In addition, in the lasso model, the variable cast_total_facebook_likes, which is also shown as not important in other methods, is removed. The coefficient estimates of some other variables shrink very close to 0, which lowers their importance in explaining the response. Overall, the lasso is probably a better choice among the above four models.

# General Additive Model

1. Introduction

We now extend the linear model to allow for certain non-linear relationship. A general additive model has the following form:

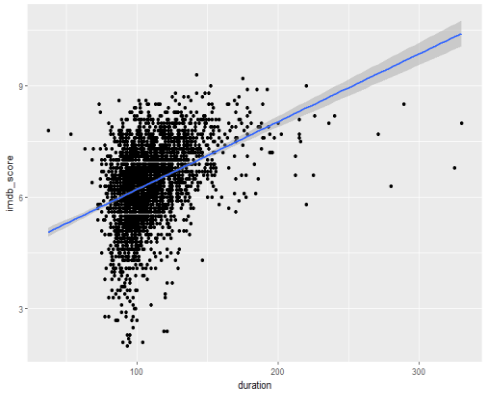
$f(x; w) = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_m\phi_m(x)$ , where $\phi_j(x): X \to R,\ j = 1, \ldots, m$ . This is still linear in $w$. $f(x; w) = w * \phi(x)$ even when $\phi$ is non-linear in the inputs $x$.

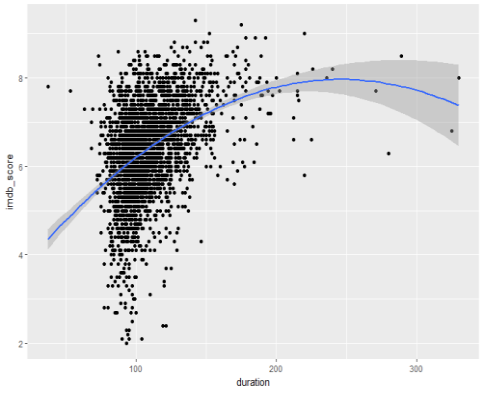
General additive models are somewhat less interpretable than linear regression, because the relationship between each predictor and the response is now modeled using a curve. Nonetheless, we deem it necessary to consider non-linear models and compare them with the linear model to find the best fit $\phi_j(x)$ .

General additive models hold the implicit assumption that the effect of changes in a predictor $x_j$ on the response $y$ is independent of the values of the other variables. We note that this assumption was violated in our case. As shown in our correlation plot, some of the predictor variables in our dataset are highly correlated with one another. It, however, could still yield some improvement over the multiple linear regression where the additivity assumption was also violated.

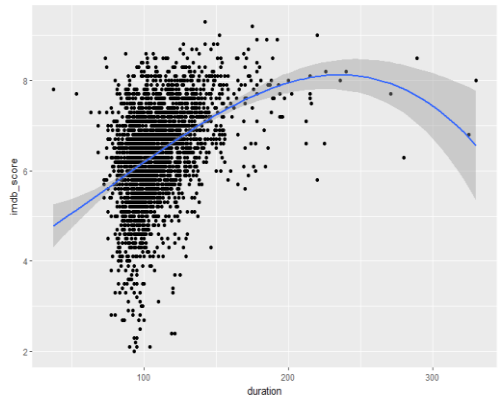
2. Non-Linear Functions

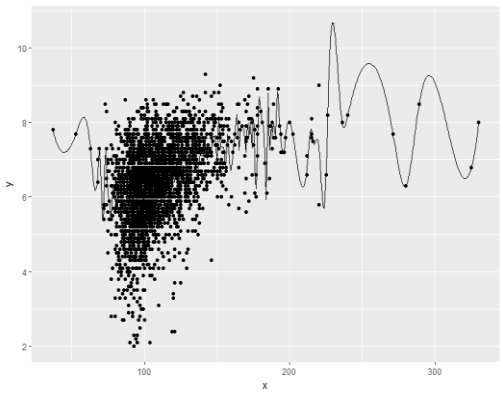
In our previous analysis of the predictor variable duration, which measures the length of a movie, we concluded that duration has a positive linear relationship with our response variable, the IMDB score. Now we will first plot curves of various degrees to compare and interpret fitness.




The points of duration seem to have a quadratic shape, suggesting that a model of the form: $Imdb\ Score = \beta_0 + \beta_1 * duration + \beta_2 * duration^2$ might be a better fit.



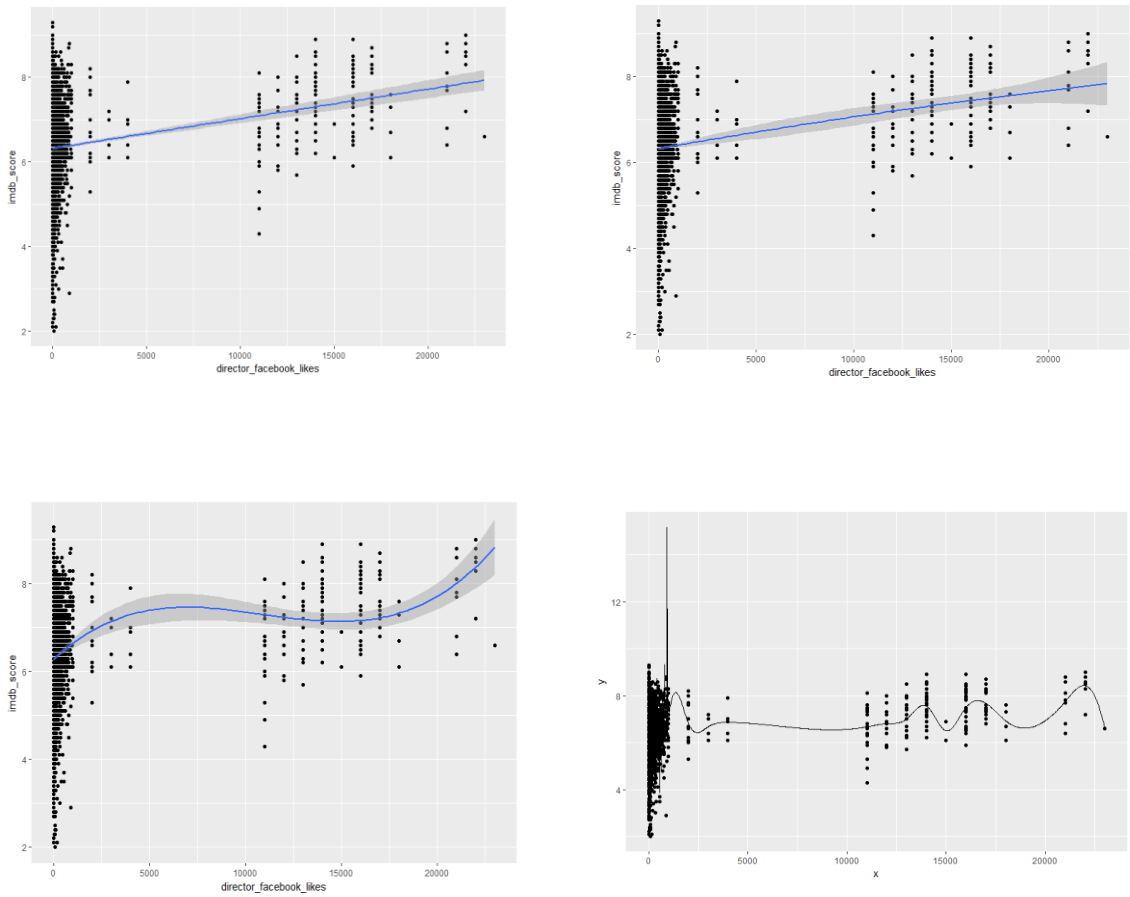




The above graphs are cubic polynomial and cubic spline respectively. The cubic curve is very similar to the quadratic one, but the curve has a wider 95% confidence interval. Likewise, the cubic spline with 30s degree of freedom seem to over-fit our data points. An Analysis of Variance (ANOVA), shown below, confirms our suspicion.

```
Analysis of Variance Table

Model 1: imdb_score ~ duration
Model 2: imdb_score ~ poly(duration, 2)
Model 3: imdb_score ~ poly(duration, 3)
Model 4: imdb_score ~ poly(duration, 4)
Model 5: imdb_score ~ poly(duration, 5)
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1   2998 2795.3
2   2997 2750.5  1    44.793 49.448 2.514e-12 ***
3   2996 2746.9  1     3.686  4.069 0.0437672 *
4   2995 2722.6  1    24.228 26.746 2.473e-07 ***
5   2994 2712.2  1    10.451 11.537 0.0006913 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> 
```

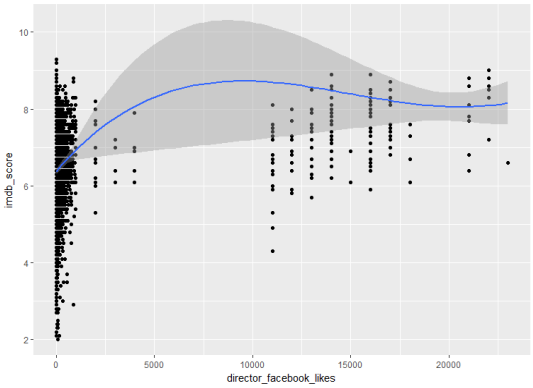
The p-value comparing linear Model 1 and quadratic Model 2 is essentially 0, indicating a linear fit is not sufficient. The p-value comparing quadratic Model 2 and cubic Model 3 is 4.3%, while degree-4 polynomial seems to be worse. Hence, either a quadratic or cubic polynomial appear to provide a reasonable fit to the data, but lower or higher order model are not justified. We proceed with the quadratic polynomial because we believe that the curve fits the data well in particular when duration is less than 200 minutes.

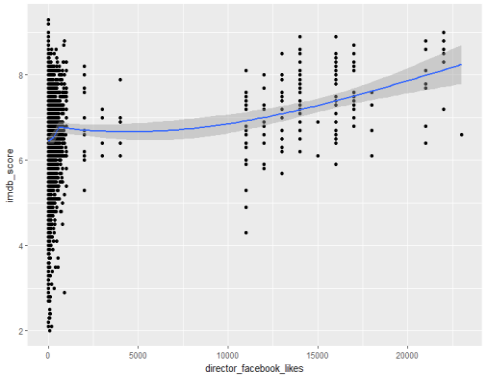
Now we conduct analysis on the predictor variable director facebook likes. The four graphs shown below are linear model, quadratic and cubic polynomial and cubic spline.




The linear and quadratic model are the same in shape, suggesting that the quadratic model has little, if any, improvement over the linear model. The cubic polynomial explains the data points as if they are oscillating but may raise the suspicion of overfitting. We note that our cubic spline in this case is overfitting the data. The curve has a spike which goes up to 14, even though the highest score possible is 10. We extend the linear and cubic polynomial to local regression and smoothing splines respectively.

The local regression, which picks a neighborhood of x to fit a regression line, does not seem to work well in our case. The 95% confidence interval is very wide due to the lack of data points of Facebook likes between 5000 to 10000. On the other hand, the smoothing spline with 3 degrees of freedom resolves the boundary issue of the cubic spline. The curve is very similar to the quadratic fit except between 0 to 1000 Facebook likes.




We use ANOVA to choose among linear model, cubic polynomial and smoothing spline.

```
Analysis of Variance Table

Model 1: imdb_score ~ director_facebook_likes
Model 2: imdb_score ~ poly(director_facebook_likes, 2)
Model 3: imdb_score ~ poly(director_facebook_likes, 3)
Model 4: imdb_score ~ s(director_facebook_likes, 3)
  Res.Df    RSS Df Sum of Sq       F    Pr(>F)
1   2998 3118.6
2   2997 3118.4  1    0.2035  0.1972     0.657
3   2996 3092.7  1   25.7487 24.9438 6.241e-07 ***
4   2998 3118.6 -2  -25.9522 12.5705 3.660e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The p-value comparing linear Model 1 and quadratic Model 2 is approximately 0.66, suggesting the quadratic or any higher order polynomial is unnecessary. To avoid overfitting, we choose the linear model for the predictor variable director facebook likes.

We perform the above procedures on all the other predictor variables and as a result construct the following general additive model.

3. Final Model

$$\begin{aligned} Imdb\ Score \sim\ & poly(duration, 2) + director\ facebook\ likes \\ & + s(num\ critic\ for\ reviews, 2) + poly(num\ user\ for\ reviews\ ,4) \\ & + s(budget, 3) + poly(cast\ total\ facebook\ likes, 4) \\ & + poly(actor\ 1\ facebook\ likes, 4) + poly(actor\ 2\ facebook\ likes, 2) \\ & + actor\ 3\ facebook\ likes + poly(movie\ facebook\ likes, 5) \\ & + poly(facenumber\ in\ poster, 2) + poly(gross, 2) + aspect\ ratio \end{aligned}$$

```
Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                       6.837e+00  1.543e-01  44.318  < 2e-16 ***
poly(duration, 2)1                1.532e+01  8.914e-01  17.182  < 2e-16 ***
poly(duration, 2)2               -4.607e+00  8.126e-01  -5.669 1.64e-08 ***
director_facebook_likes           2.283e-05  5.013e-06   4.555 5.56e-06 ***
s(num_critic_for_reviews, 2)      1.350e-03  2.690e-04   5.018 5.68e-07 ***
poly(num_user_for_reviews, 4)1    3.234e+00  1.145e+00   2.824 0.004787 **
poly(num_user_for_reviews, 4)2   -1.725e+00  9.166e-01  -1.881 0.060058 .
poly(num_user_for_reviews, 4)3    2.976e+00  8.275e-01   3.596 0.000331 ***
poly(num_user_for_reviews, 4)4   -1.335e+00  8.067e-01  -1.655 0.098038 .
s(budget, 3)                     -6.396e-09  5.248e-10 -12.188  < 2e-16 ***
poly(cast_total_facebook_likes, 4)1 -4.362e+01  1.152e+01  -3.787 0.000157 ***
poly(cast_total_facebook_likes, 4)2  1.058e+01  3.017e+00   3.506 0.000465 ***
poly(cast_total_facebook_likes, 4)3 -2.496e+00  2.172e+00  -1.149 0.250593
poly(cast_total_facebook_likes, 4)4  1.100e+00  1.763e+00   0.624 0.532725
poly(actor_1_facebook_likes, 4)1     3.775e+01  8.192e+00   4.609 4.30e-06 ***
poly(actor_1_facebook_likes, 4)2    -1.201e+01  2.859e+00  -4.200 2.79e-05 ***
poly(actor_1_facebook_likes, 4)3     5.321e+00  1.900e+00   2.801 0.005141 **
poly(actor_1_facebook_likes, 4)4    -1.546e+00  1.500e+00  -1.031 0.302838
poly(actor_2_facebook_likes, 2)1     1.198e+01  3.687e+00   3.250 0.001173 **
poly(actor_2_facebook_likes, 2)2    -9.047e-01  1.292e+00  -0.700 0.483924
actor_3_facebook_likes               2.325e-05  2.473e-05   0.940 0.347247
poly(movie_facebook_likes, 5)1       2.460e+00  1.232e+00   1.997 0.045915 *
poly(movie_facebook_likes, 5)2      -1.838e+00  8.616e-01  -2.133 0.033015 *
poly(movie_facebook_likes, 5)3       2.999e+00  8.203e-01   3.656 0.000263 ***
poly(movie_facebook_likes, 5)4      -2.088e+00  8.076e-01  -2.585 0.009812 **
poly(movie_facebook_likes, 5)5       2.465e+00  8.021e-01   3.073 0.002149 **
poly(facenumber_in_poster, 2)1      -3.623e+00  7.996e-01  -4.531 6.21e-06 ***
poly(facenumber_in_poster, 2)2       6.562e-01  7.949e-01   0.826 0.409132
poly(gross, 2)1                      1.245e+01  1.136e+00  10.959  < 2e-16 ***
poly(gross, 2)2                     -3.884e+00  8.740e-01  -4.444 9.30e-06 ***
aspect_ratio                        -1.908e-01  7.012e-02  -2.721 0.006563 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.782 on 2069 degrees of freedom
Multiple R-squared:  0.4286,    Adjusted R-squared:  0.4203
F-statistic: 51.73 on 30 and 2069 DF,  p-value: < 2.2e-16
```

The coefficients themselves as discussed earlier are hard to interpret, and most of them are statistically significant with exceptions on cast total facebook likes and actor 1 facebook likes, suggesting overfitting of these two variables.

We already conducted curve fitting and ANOVA of individual non-linear function to avoid overfitting. But overfitting still occurs in our general additive model. This phenomenon can be attributed to multicollinearity. For example, the cast total facebook likes is highly correlated with actor 1, 2, 3 facebook likes. One might also say that the cast total facebook likes includes some of the actor facebook likes. Due to the nature of our dataset, our general additive model might unwittingly overfit the movie dataset.

The ANOVA for our general additive model, shown below, indicates that the three predictor variables actor 1, 2, 3 contribute little to the residual sum of squares. And as a result, the fourth order polynomial on actor 1 facebook likes and quadratic model on actor 2 facebook likes might overfit the dataset.

```
(Dispersion Parameter for gaussian family taken to be 0.6264)

    Null Deviance: 2571.572 on 2399 degrees of freedom
Residual Deviance: 1482.069 on 2366 degrees of freedom
AIC: 5724.035

Number of Local Scoring Iterations: 2

Anova for Parametric Effects
                                  Df  Sum Sq Mean Sq  F value    Pr(>F)
s(num_critic_for_reviews, 2)       1  390.79  390.79 623.8695 < 2.2e-16 ***
poly(num_user_for_reviews, 4)      4  140.25   35.06  55.9761 < 2.2e-16 ***
poly(duration, 2)                  2  243.76  121.88 194.5727 < 2.2e-16 ***
s(budget, 3)                       1   70.17   70.17 112.0126 < 2.2e-16 ***
director_facebook_likes            1   25.37   25.37  40.5073 2.345e-10 ***
poly(cast_total_facebook_likes, 4) 4   24.99    6.25   9.9745 5.281e-08 ***
poly(actor_1_facebook_likes, 4)    4   14.34    3.59   5.7241 0.0001391 ***
poly(actor_2_facebook_likes, 2)    2    8.65    4.32   6.9026 0.0010255 **
actor_3_facebook_likes             1    0.64    0.64   1.0191 0.3128344
poly(movie_facebook_likes, 5)      5   31.75    6.35  10.1357 1.276e-09 ***
poly(facenumber_in_poster, 2)      2   11.46    5.73   9.1442 0.0001107 ***
poly(gross, 2)                     2   99.81   49.90  79.6659 < 2.2e-16 ***
aspect_ratio                       1    4.89    4.89   7.8101 0.0052374 **
Residuals                       2366 1482.07    0.63
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Anova for Nonparametric Effects
                                  Npar Df  Npar F     Pr(F)
(Intercept)
s(num_critic_for_reviews, 2)            1  8.3404  0.003912 **
poly(num_user_for_reviews, 4)
poly(duration, 2)
s(budget, 3)                            2 20.8647 1.041e-09 ***
director_facebook_likes
poly(cast_total_facebook_likes, 4)
poly(actor_1_facebook_likes, 4)
poly(actor_2_facebook_likes, 2)
actor_3_facebook_likes
poly(movie_facebook_likes, 5)
poly(facenumber_in_poster, 2)
poly(gross, 2)
aspect_ratio
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We want to compare the saturated linear model with our general additive model. We conduct a 10 fold cross validation of these two models on our training set and test them on our test set. The training set comprises 70% of total dataset selected at random, the test set 30%.

Below is a table which compares the mean square of errors (MSE) of linear full model and general additive model on training and test error.

| MSE | Linear Model (Full Model) | General Additive Model |
|---|---|---|
| Training | 0.735 | 0.698 |
| Test | 0.711 | 0.734 |

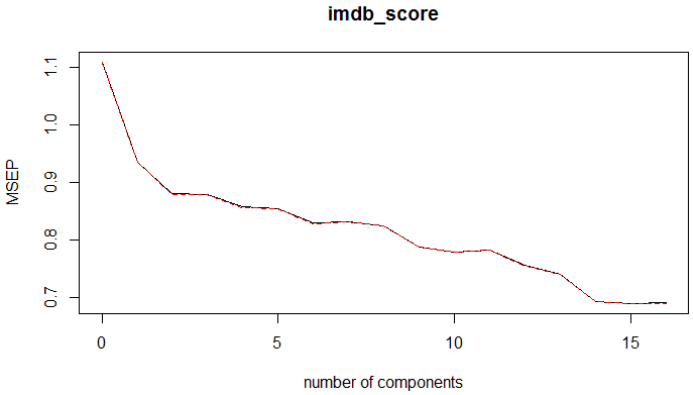
We conclude that our general additive model has a slightly smaller training error than the saturated linear model but a slightly larger test error. It does not seem to overfit our dataset in particular although it raises certain suspicions of it in our analysis of coefficients and variance. Hence, it is a valid alternative to interpret our dataset which allows for more flexibility than our linear model.

## Principal Components Regression

The principal components regression (PCR) approach involves constructing principal components regression the first M principal components, $Z_1,...,Z_M$, and then using these components as the predictors in a linear regression model that is fit using least squares. A small number of principal components suffice to explain most of the variability in the data, as well as the relationship with

the response. We assume that the directions in which $X_1,\ldots, X_p$ show the most variation are the directions that are associated with Y.

I randomly set our data to 70% training set and 30% testing set. We see that the smallest 10 fold cross-validation error occurs with 15 components.

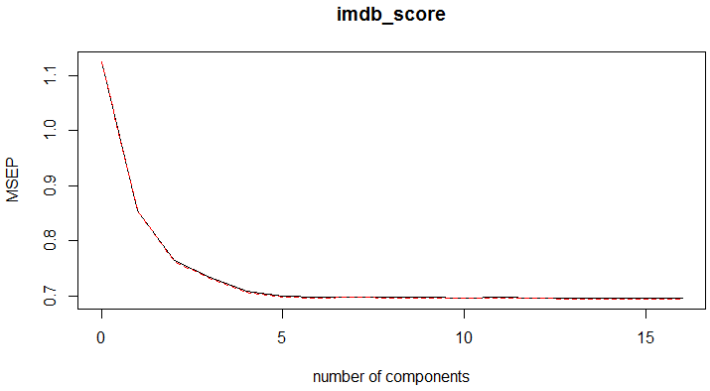



```
Fit method: svdpc
Number of components considered: 16

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
CV           1.053   0.9668   0.9379   0.9372   0.9255   0.9237   0.9103   0.9117   0.9077
adjCV        1.053   0.9663   0.9376   0.9371   0.9251   0.9236   0.9100   0.9118   0.9074
       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps  16 comps
CV      0.8875    0.8825    0.8843    0.8688    0.8604    0.8323    0.8304    0.8305
adjCV   0.8870    0.8820    0.8842    0.8684    0.8600    0.8318    0.8299    0.8299

TRAINING: % variance explained
           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
X            29.02    42.34    54.69    61.29    67.71    73.74    79.20    84.41    88.80
imdb_score   16.33    20.87    20.87    22.98    23.42    25.49    25.52    26.65    30.09
           10 comps  11 comps  12 comps  13 comps  14 comps  15 comps  16 comps
X             92.67     95.35     97.50     98.93     99.99    100.00    100.00
imdb_score    30.99     30.99     33.04     34.35     38.67     39.05     39.05
```

We now perform PCR on the training data and evaluate its test set performance, and notice that the relative low cross-validation error occurs when 6 component are used. Finally, we fit PCR on the full data set, using 6 components identified by cross-validation. We compute the test MSE as follows. MSE of 70% training dataset is 0.8468189, and MSE of 30% testing dataset is 0.809612. This test set MSE is competitive with the results obtained using ridge and the lasso. However, as a result of the way PCR is implemented, the final model is more difficult to interpret because it does not perform any kind of variable selection or even directly produce coefficient estimates.

## Partial Least Squares

PLS is also a dimension reduction method, which first identifies squares a new set of features $Z_1,\ldots,Z_M$ that are linear combinations of the original features, and then fits a linear model by least squares using these M new features. The lowest cross-validation error occurs when only 6 partial least squares directions are used.




```
Fit method: kernelpls
Number of components considered: 16

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
CV           1.061   0.9247   0.8738   0.8562   0.8412   0.8358   0.8355   0.8355   0.8351
adjCV        1.061   0.9244   0.8735   0.8553   0.8404   0.8350   0.8347   0.8348   0.8344
       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps  16 comps
CV      0.8351    0.8345    0.8352    0.8345    0.8341    0.8339    0.8338    0.8338
adjCV   0.8344    0.8339    0.8346    0.8338    0.8334    0.8332    0.8331    0.8331

TRAINING: % variance explained
           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps
X            29.13    40.51    48.29    52.01    55.69    61.00    69.70    75.73    79.42
imdb_score   24.54    33.01    36.67    38.78    39.47    39.56    39.59    39.60    39.61
           10 comps  11 comps  12 comps  13 comps  14 comps  15 comps  16 comps
X             81.86     86.43     88.35     92.38     97.50    100.00    100.22
imdb_score    39.63     39.68     39.83     39.86     39.86     39.86     39.86
```

Using the training data, we can plot the predicted imdb_score from the training set to the actual imdb_score. Finally, we perform PLS using the full data set with 6 components identified by cross-validation. MSE of 70% training dataset is 0.6947274, and MSE of 30% testing dataset is 0.6857737.

With 6 components, we notice that the percentage of variance in x that using PLS is less than using PCR. And the percentage of variance in imdb_score that using PLS is more than using the PCR. This is because PCR only attempts to maximize the amount of variance explained in the predictors, while PLS searches for directions that explain variance in both the predictors and the response. The PLS direction does not fit the predictors as closely as does PCA, but it does a better job explaining the response.

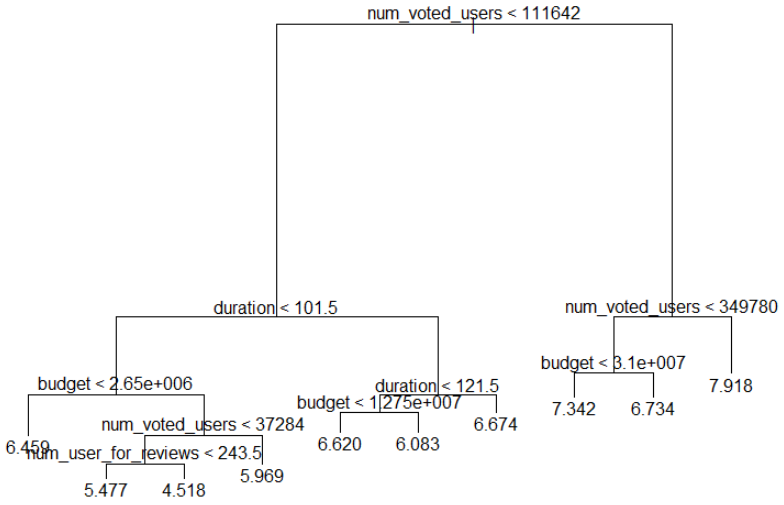
# Regression Tree

Tree-based method is another non-linear method. It has a number of advantages over the more classical approaches we talked about before. Trees are very easy to explain to people and are closely mirror human decision-making process. We could easily handle qualitative predictors without creating dummy variables. Another advantage is that there is no need to scale variables before building a tree.

This involves stratifying or segmenting the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean or the mode of training observations in the region to which it belongs.

Since the dependent variable is *imdb_score*, which is a continuous variable, we use regression tree to build our model. In order to divide the predictor space into distinct and non-overlapping regions $R_1, R_2, \ldots, R_J$, we use a top-down greedy approach called recursive binary splitting. At each split, we select a predictor $X_j$ and a cut point $s$ such that splitting the predictor space into regions $\{X|X_j<s\}$ and $\{X|X_j\geq s\}$ leads to the greatest possible reduction in $RSS = \sum_{j=1}^{J}\sum_{i\in R_j}(y_i - \hat{y}_{R_j})^2$.

Below is a regression tree.



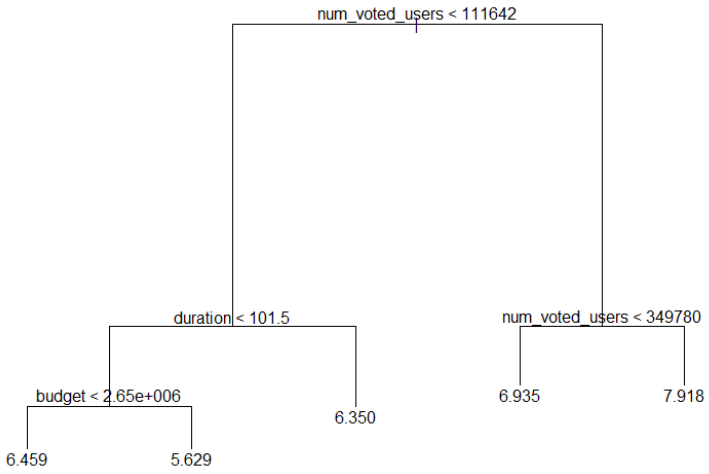
The regression tree above consists of a series of splitting rules, starting at the top of the tree. The top split assigns observations having *num_voted_users<111642* to the left branch and the others to the right branch. For the right branch, we further split it with variable *num_voted_users* and *budget*. There is a predicted *imdb_score* for each region, so the predicted values are discrete variables.

After dividing the original data set into training set and test set, we calculate the MSE on training set and test set. They are 0.6503837 and 0.7050944.

# Tree Pruning

A very complex decision tree may lead to overfitting problem. The process described above may produce good predictions on training set, but poor predictions on test set. In order to solve that problem, we could build a smaller tree with fewer splits. Although there might be a little bias, the variance would reduce significantly.

We could first grow a very large tree $T_0$, and then prune it back in order to obtain a subtree. It is inefficient to consider each possible subtree. Instead, we use *cost complexity pruning* - also known as *weakest link pruning* method. Similar as *Lasso*, here we introduce a nonnegative tuning parameter $\alpha$ . For each value of $\alpha$ there corresponds a subtree $T \subset T_0$ such that $\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$ is as small as possible. Here $|T|$ indicates the number of terminal nodes of tree *T*. In *R*, we could choose the depth of a pruned tree. Figure ? is an example of the original tree after pruning.




Pruned Tree

This tree is much simpler than the former tree. All observations fall into five groups.

However, there is an obvious disadvantage of the tree we build. It can be very non-robust. A small change in the data can cause a large change in the final estimated tree. By aggregating many regression trees, using methods like *bagging*, *random forests*, and *boosting*, the predictive performance of trees can be substantially improved.

## Bagging

Recall central limit theorem, by averaging a set of observations, we could reduce variance and improve accuracy. *Bagging* is a general-purpose procedure for reducing the variance of regression trees. Like bootstrap, we could take repeated samples from the single training data set. We simply construct *B* regression trees using *B* bootstrapped training sets, and average the resulting predictions. Each individual tree has high variance but low bias. Averaging these *B* trees reduces the variance.

The MSE of training set is 0.008620443 and the MSE of test set is 0.5027234. There is a remarkable reduction in MSE of test set.

## Random Forests

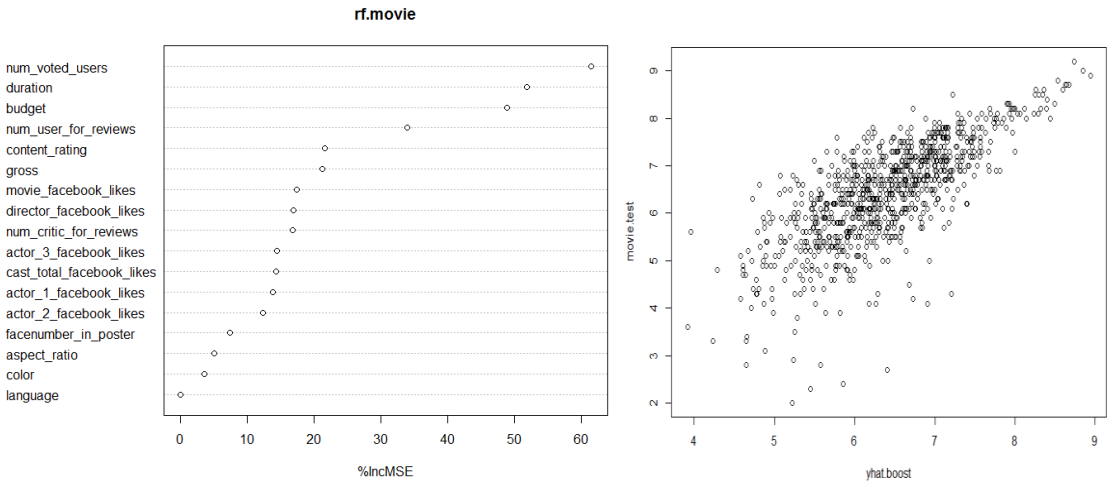
*Random forests* provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. Unlike bagging, the number of predictors considered at each split is approximately equal to the one third of the total number of predictors. In order words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors.

Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors. Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence the predictions from the bagged trees will be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. Random forests overcome this problem by forcing each split to consider only a subset of the predictors, thereby making the average of the resulting trees less variable and hence more reliable.

There are total of 18 variables in our data set, and the number of variables randomly sampled as candidates at each split is 6. The MSE of training set is 0.09154073 and the MSE of test set is 0.4990712. The performance of *random forest* is even better than *bagging*.

Besides, we can obtain an overall summary of the importance of each predictor using the RSS. In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all *B* trees. A large value indicates an important predictor. The variable importance rank is shown below.

Random forests are not very sensitive to the specific hyper-parameters used. They use a large number of trees to ensure this.




# Boosting

Boosting works in a similar way as random forests, except that the trees are grown sequentially: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set. Boosting involves combining a large number of decision trees. We fit a decision tree to the residuals from the model. That is, we fit a tree using the current residuals, rather than the outcome Y, as the response. We then add this new decision tree into the fitted function in order to update the residuals.

Below is a table that compares the MSE of the tree methods. The Bagging, Random Forests, and Boosting has more than 30% improvement over our linear regressions and general additive model.

| MSE | Unpruned Tree | Bagging | Random Forests | Boosting |
|---|---|---|---|---|
| Training | 0.6503837 | 0.08620443 | 0.09154073 | 0.2741793 |
| Test | 0.7050944 | 0.5027234 | 0.4990712 | 0.5009072 |