PREDICTING ONLINE NEWS POPULARITY

Springboard Capstone Project
Technical Document

Tushar Prakash

Background

This project deals with popularity of online news. The internet age gives us both the need and opportunity to analyze factors that determine the number of shares a news article gets on the social media and predict the shares based on those factors for new news articles.
The data for this project is provided comes from the popular news sharing site Mashable.com. Mashable presents news articles from various new and media outlets on many different topics. It even gives the number of shares on its news articles to indicate the popularity of the article. The goal that Mashable proposes for this data set is to predict whether the number of shares will be greater than 1400 or less than that number.

In this project I will explore why Mashable proposes this goal as a binary classification problem, and why it draws a boundary on the number 1400. I will also explore the feasibility of a more granular classification, which would more realistically reflect the share categories that Mashable news article actually fall into.

The data set provided by Mashable was taken from UCI machine learning laboratory. For more information on the data, please check this link.

1. Data Description

The data set obtained from UCI Machine learning laboratory contains 39,644 records with 59 quantified features related to the particular news articles. In addition, there is one column for the link to the news article itself, and one for the dependent variable, i.e. the number of shares the news article gets.

The data set is described below:

0. url	URL of the article
1. timedelta	Days between the article publication and the dataset acquisition
2. n_tokens_title	Number of words in the title
3. n_tokens_content	Number of words in the content
4. n_unique_tokens	Rate of unique words in the content
5. n_non_stop_words	Rate of non-stop words in the content
6. n_non_stop_unique_tokens	Rate of unique non-stop words in the content
7. num_hrefs	Number of links
8. num_self_hrefs	Number of links to other articles published by Mashable
9. num_imgs	Number of images
10. num_videos	Number of videos
11. average_token_length	Average length of the words in the content
12. num_keywords	Number of keywords in the metadata
13. data_channel_is_lifestyle	Is data channel 'Lifestyle'?
14. data_channel_is_entertainment	Is data channel 'Entertainment'?
15. data_channel_is_bus	Is data channel 'Business'?
16. data_channel_is_socmed	Is data channel 'Social Media'?

17. data_channel_is_tech	Is data channel 'Tech'?
18. data_channel_is_world	Is data channel 'World'?
19. kw_min_min	Worst keyword (min. shares)
20. kw_max_min	Worst keyword (max. shares)
21. kw_avg_min	Worst keyword (avg. shares)
22. kw_min_max	Best keyword (min. shares)
23. kw_max_max	Best keyword (max. shares)
24. kw_avg_max	Best keyword (avg. shares)
25. kw_min_avg	Avg. keyword (min. shares)
26. kw_max_avg	Avg. keyword (max. shares)
27. kw_avg_avg	Avg. keyword (avg. shares)
28. self_reference_min_shares	Min. shares of referenced articles in Mashable
29. self_reference_max_shares	Max. shares of referenced articles in Mashable
30. self_reference_avg_sharess	Avg. shares of referenced articles in Mashable
31. weekday_is_monday	Was the article published on a Monday?
32. weekday_is_tuesday	Was the article published on a Tuesday?
33. weekday_is_wednesday	Was the article published on a Wednesday?
34. weekday_is_thursday	Was the article published on a Thursday?
35. weekday_is_friday	Was the article published on a Friday?
36. weekday_is_saturday	Was the article published on a Saturday?
37. weekday_is_sunday	Was the article published on a Sunday?
38. is_weekend	Was the article published on the weekend?
39. LDA_00	Closeness to LDA topic 0
40. LDA_01	Closeness to LDA topic 1
41. LDA_02	Closeness to LDA topic 2
42. LDA_03	Closeness to LDA topic 3
43. LDA_04	Closeness to LDA topic 4
44. global_subjectivity	Text subjectivity
45. global_sentiment_polarity	Text sentiment polarity
46. global_rate_positive_words	Rate of positive words in the content
47. global_rate_negative_words	Rate of negative words in the content
48. rate_positive_words	Rate of positive words among non-neutral tokens
49. rate_negative_words	Rate of negative words among non-neutral tokens
50. avg_positive_polarity	Avg. polarity of positive words
51. min_positive_polarity	Min. polarity of positive words
52. max_positive_polarity	Max. polarity of positive words
53. avg_negative_polarity	Avg. polarity of negative words
54. min_negative_polarity	Min. polarity of negative words
55. max_negative_polarity	Max. polarity of negative words
56. title_subjectivity	Title subjectivity
57. title_sentiment_polarity	Title polarity

58. abs_title_subjectivity	Absolute subjectivity level
59. abs_title_sentiment_polarity	Absolute polarity level
60. shares	Number of shares (target)

2. Feature Selection

Before doing any data analysis on the data for reducing dimensions, and in effect selecting the most significant features, we need to do some data massaging.

2.1 Capping

The number of shares varied from less than 100 to more than 80,000. This was not a uniform distribution. As can be seen below in the histogram, it is heavily skewed towards the left. I removed the outliers by capping the data below 5% quantile to the 5% quantile value, and data above 95% quantile to the 95% quantile value. I did this for each of the columns accept the first one which contains the actual news link. It is evident that after capping, the distribution becomes comparatively smooth.

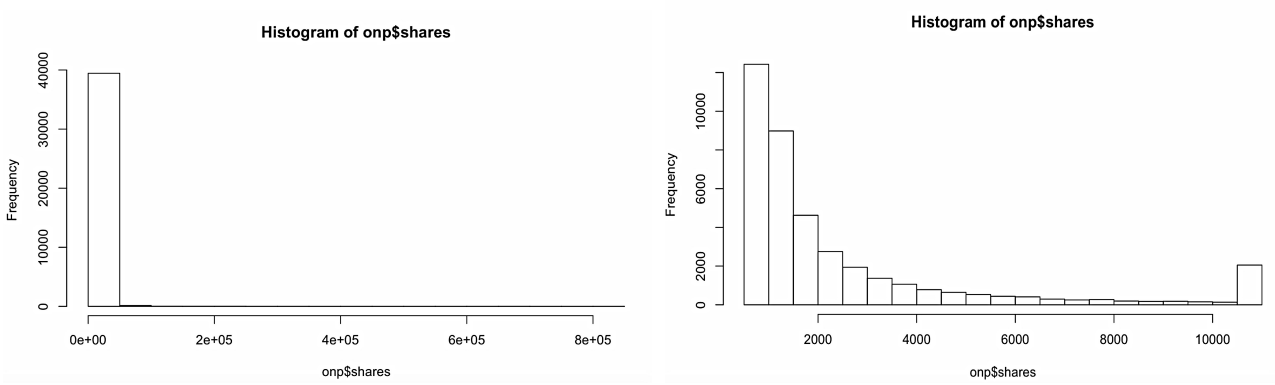

Figure 1 Histograms of 'Shares' before and after Capping

2.2 Factoring

I converted all the categorical variables in the original data set to factors.

2.3 Scaling

I centered and scaled all the continuous variables to bring all these variables on the same scale and make model coefficients intuitive and interpretable. The following variables were scaled:

1) self_reference_min_shares
2) self_reference_avg_sharess
3) self_reference_max_shares
4) kw_avg_min
5) kw_avg_max
6) kw_avg_avg

2.4 Data Analysis

We need to select only the significant features of the model to reduce complexity and multicollinearity in the models. Principal Component Analysis (PCA) or Multi Factor Analysis (MFA) did not seem particularly suitable as feature selection techniques as my variable set contains a mixture of binary factors and continuous variables. Given the constraints, I took a more manual approach to feature selection. I used the rcorr() function to generate the correlation matrix along with the significance levels of each correlation. I flattened this matrix to get all the unique variable pairs and the correlations between them in rows. I started with studying only a subset of the variable pairs, which had absolute correlations greater than 10% with significance levels of greater than 95% (P-value < 0.05). From each of these variable pairs, I select the variable that has a greater correlation with the dependent variable (no. of shares).

The analysis described above does not yield clean solutions. This is because the above selection process has to deal with contests between pairs of variables that are not mutually exclusive. To give an example, *avg_positive_polarity* is a choice variable between many comparisons, but between avg_positive_polarity and *abs_title_sentiment_polarity*, *abs_title_sentiment_polarity* gets chosen. So between *avg_positive_polarity* and *abs_title_sentiment_polarity*, which one should get chosen?

To resolve this problem I devised a strategy that I have described schematically on the following page. We start with first rejecting any winning variable that has less than 1% correlation with the dependent variable. I call a particular variable a 'winning' variable here, if it's the chosen one within a pair. As described above, a winning variable, say variable **A**, might be the looser in another contest with variable **B**. We want to know, with justifiable reasons, whether we should still select variable A and reject B, or select the winning variable B and reject the previous winner A, or select both.

Given both have greater than 1% correlation with the dependent variable, say Y, we check whether A had greater number of wins than B. We also check which among the two has a greater correlation with **Y**. If for these two tests, there's a clear winner then we take that variable. If not, we check whether there's a significant correlation between these two variables themselves. If so, we select the one with greater number of wins and reject the one with higher correlation with the dependent variable. The rationale behind this is thus:

1. If two variables have high correlation between themselves, they will likely have comparable correlation with dependent variable. Choosing both will introduce multicollinearity and complexity into the model
2. We choose the variable with higher number of wins because, given 1, we get to simplify the model without suffering from a significant loss of variance in the model.

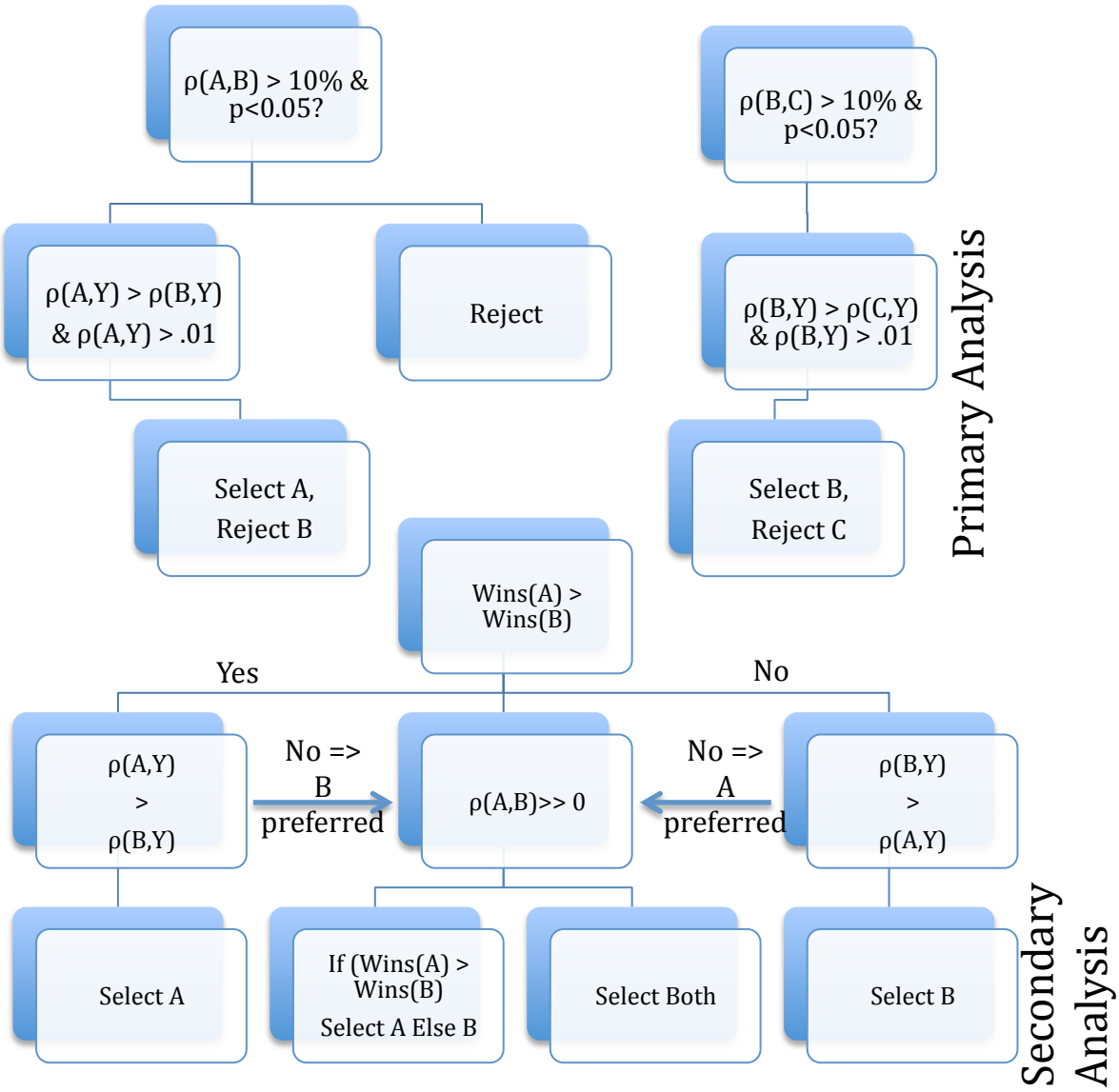

Figure 2 Dimentionality Reduction Scheme using correlation analysis

The last scenario is that there is no clear winner between A and B, and nor do they have a significant correlation between themselves. In this case we choose both. The contest between *avg_positive_polarity* and *abs_title_sentiment_polarity* fall into this exact category as described below.

A	B	C	D	E	F	G	H
row	column	cor	p	rcor	ccor	selection	selection_Cc
global_rate_positive_words	abs_title_sentiment_polarity	0.11969794	0	3.13682854	5.36513403	abs_title_sentiment_polarity	5.365134031
avg_positive_polarity	abs_title_sentiment_polarity	0.1173523	0	5.16295694	5.36513403	abs_title_sentiment_polarity	5.365134031
title_sentiment_polarity	abs_title_sentiment_polarity	0.48374337	0	4.23559248	5.36513403	abs_title_sentiment_polarity	5.365134031
abs_title_subjectivity	abs_title_sentiment_polarity	-0.4730971	0	0.03870663	5.36513403	abs_title_sentiment_polarity	5.365134031
n_tokens_content	avg_positive_polarity	0.10197602	0	0.06538966	5.16295694	avg_positive_polarity	5.162956938
n_unique_tokens	avg_positive_polarity	0.13929011	0	-1.3243237	5.16295694	avg_positive_polarity	5.162956938
n_non_stop_unique_tokens	avg_positive_polarity	0.13682675	0	-4.4260018	5.16295694	avg_positive_polarity	5.162956938
global_sentiment_polarity	avg_positive_polarity	0.51131201	0	3.62135507	5.16295694	avg_positive_polarity	5.162956938
global_rate_positive_words	avg_positive_polarity	0.24180487	0	3.13682854	5.16295694	avg_positive_polarity	5.162956938
global_rate_negative_words	avg_positive_polarity	0.15009199	0	0.3735878	5.16295694	avg_positive_polarity	5.162956938
rate_positive_words	avg_positive_polarity	0.18922739	0	0.31772605	5.16295694	avg_positive_polarity	5.162956938
avg_positive_polarity	min_positive_polarity	0.35672152	0	5.16295694	-0.3094281	avg_positive_polarity	5.162956938
avg_positive_polarity	max_positive_polarity	0.61961991	0	5.16295694	4.36378308	avg_positive_polarity	5.162956938
avg_positive_polarity	avg_negative_polarity	-0.2205654	0	5.16295694	-3.7841301	avg_positive_polarity	5.162956938
avg_positive_polarity	min_negative_polarity	-0.1786767	0	5.16295694	-2.6955815	avg_positive_polarity	5.162956938
avg_positive_polarity	max_negative_polarity	-0.1071208	0	5.16295694	-1.1717788	avg_positive_polarity	5.162956938

Figure 3 CSV file generated for data analysis

According to the scheme defined above, we observe that between avg_positive_polarity and abs_title_sentiment_polarity, avg_positive_polarity has the double the number of wins but abs_title_sentiment_polarity has double the correlation with number of shares. So there's no clear choice. We then see whether correlation between avg_positive_polarity and abs_title_sentiment_polarity is significant. It is only 10%, our lower limit to even consider correlations between variables. Hence, we choose both and continue our analysis.

In such an analysis scheme, one of the two variables may get eliminated in another round of analysis. Following this scheme, I was able to reduce the significant features to 22 out of 60. On running these parameters in linear models, I eliminated another 3, which were not significant, leaving the set of selected features down to 20.

The following features made the final cut:

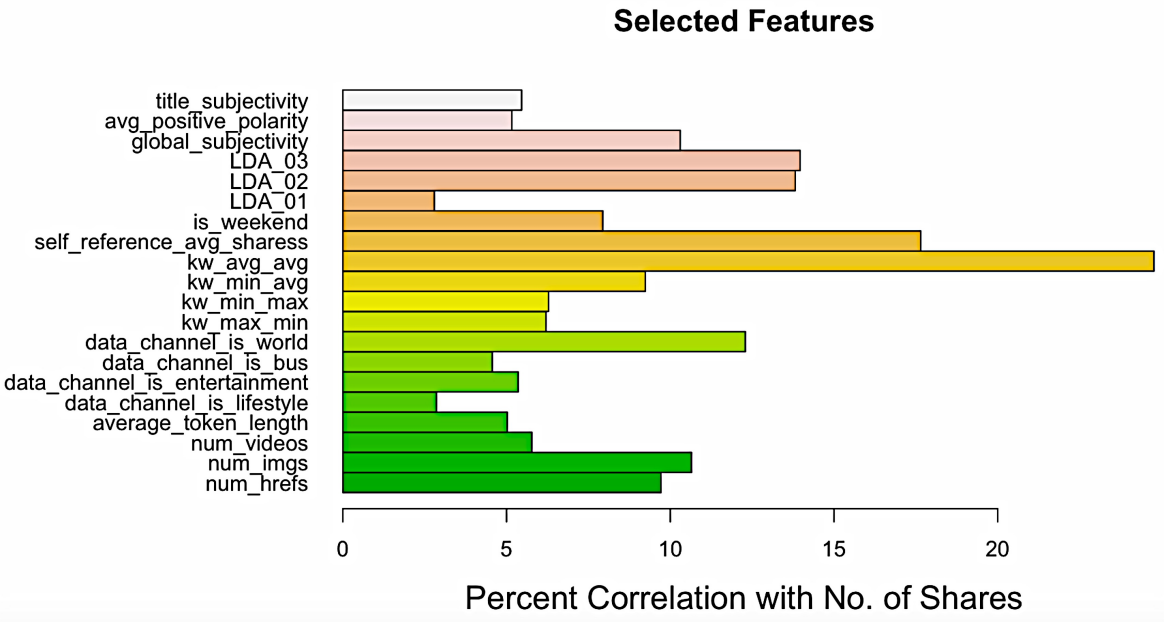

Figure 4 Correlations of 'Shares' with selected features

3. Pre-Model Data Preparation

3.1 Scaling the dependent variable

In addition to the data preparation done before data analysis, I introduced a new variable "scaledy", which is nothing but scaled dependent variable, i.e. the number of shares. Since number of shares is the dependent variable, we want to preserve the original variable for further use and keep it separate from it's scaled version.

3.2 Classification

I intended to study this problem through both regression and classification. On an analysis of deciles of the number of shares, we find that 5th decile corresponds to exactly 1400 shares. It is now clear that original problem wants us to be able to discriminate between that two halves of it's observed number of shares.

```
>quantile(onp$shares, probs = c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1))
 0%   20%   30%   40%   50%   60%   70%   80%   90%  100%
 08.0  870.6 1000.0 1200.0 1400.0 1800.0 2300.0 3400.0 6200.0 10800.0
```

Hence I introduced two new variables for two-way classification. One is a binary classification between shares less than or equal to 1400 and greater than 1400. I do another 30-30-20-20 classification based on the above quantiles. Hence the 4-class classification is between shares up to a 1000, greater than 1000 and less than or equal to 1800, greater than 1800 and less than or equal to 3400, and greater than 3400. I present the distribution of data in the two classes as below:

```
  > table(onp$classes)
1    2    3    4
12424 12020  7392  7808

> table(onp$classes2)
   0    1
20082 19562
```

3.3 Factoring the classes

To be able to tell the regression or classification models that the two variables constructed above are indeed classes, I converted them to factors.

3.4 Splitting

I split the data into training and test sets on a 75%-25% data split.

4. Models and Results

4.1 Linear Regression

I started with a linear regression model, using the selected features. I used *scaledy* as the dependent variable. The R-squared from this model was a low 0.099.

```
> summary(model1.lr)

Call:
lm(formula = scaledy ~ kw_avg_avg + self_reference_avg_sharess +
    LDA_03 + data_channel_is_world + num_imgs + global_subjectivity +
    num_hrefs + kw_min_avg + is_weekend + kw_min_max + kw_max_min +
    num_videos + data_channel_is_socmed + title_subjectivity +
    data_channel_is_entertainment + avg_positive_polarity + average_token_length +
    data_channel_is_lifestyle + LDA_01 + LDA_02 + LDA_04 + data_channel_is_bus,
    data = TrainONP)

Residuals:
   Min     1Q  Median     3Q    Max
-1.8275 -0.5159 -0.2791  0.0909  3.6788

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  6.329e-01  1.149e-01   5.509 3.63e-08 ***
kw_avg_avg                   2.032e-01  8.911e-03  22.804  < 2e-16 ***
self_reference_avg_sharess   1.037e-01  5.771e-03  17.969  < 2e-16 ***
LDA_03                      -2.617e-01  4.293e-02  -6.096 1.10e-09 ***
data_channel_is_world1      -5.952e-02  2.676e-02  -2.224 0.026123 *
num_imgs                     6.517e-03  1.115e-03   5.846 5.10e-09 ***
global_subjectivity          3.358e-01  7.801e-02   4.304 1.68e-05 ***
num_hrefs                    3.766e-03  8.128e-04   4.634 3.61e-06 ***
kw_min_avg                  -4.082e-05  7.820e-06  -5.220 1.81e-07 ***
is_weekend1                  1.821e-01  1.642e-02  11.088  < 2e-16 ***
kw_min_max                  -3.596e-07  6.212e-07  -0.579 0.562687
kw_max_min                   2.493e-05  9.257e-06   2.693 0.007078 **
num_videos                   1.280e-02  4.202e-03   3.045 0.002327 **
data_channel_is_socmed1      8.104e-02  2.825e-02   2.869 0.004125 **
title_subjectivity           5.203e-02  1.733e-02   3.003 0.002678 **
data_channel_is_entertainment1 -1.823e-01  2.092e-02  -8.714  < 2e-16 ***
avg_positive_polarity        -2.270e-01  8.109e-02  -2.800 0.005120 **
```

```
average_token_length        -1.081e-01  2.270e-02  -4.762 1.92e-06 ***
data_channel_is_lifestyle1   -9.599e-02  2.677e-02  -3.586 0.000337 ***
LDA_01                      -2.751e-01  5.087e-02  -5.407 6.45e-08 ***
LDA_02                      -3.553e-01  4.745e-02  -7.489 7.15e-14 ***
LDA_04                      -1.547e-01  4.078e-02  -3.793 0.000149 ***
data_channel_is_bus1        -2.077e-01  2.816e-02  -7.376 1.67e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.947 on 29710 degrees of freedom
Multiple R-squared:  0.0971,          Adjusted R-squared:  0.09643
F-statistic: 145.2 on 22 and 29710 DF,  p-value: < 2.2e-16
```

Moreover, we can also observe from the correlogram of the residuals that theirs is no correlation present between the residuals, and hence the residuals are truly independent and unbiased.

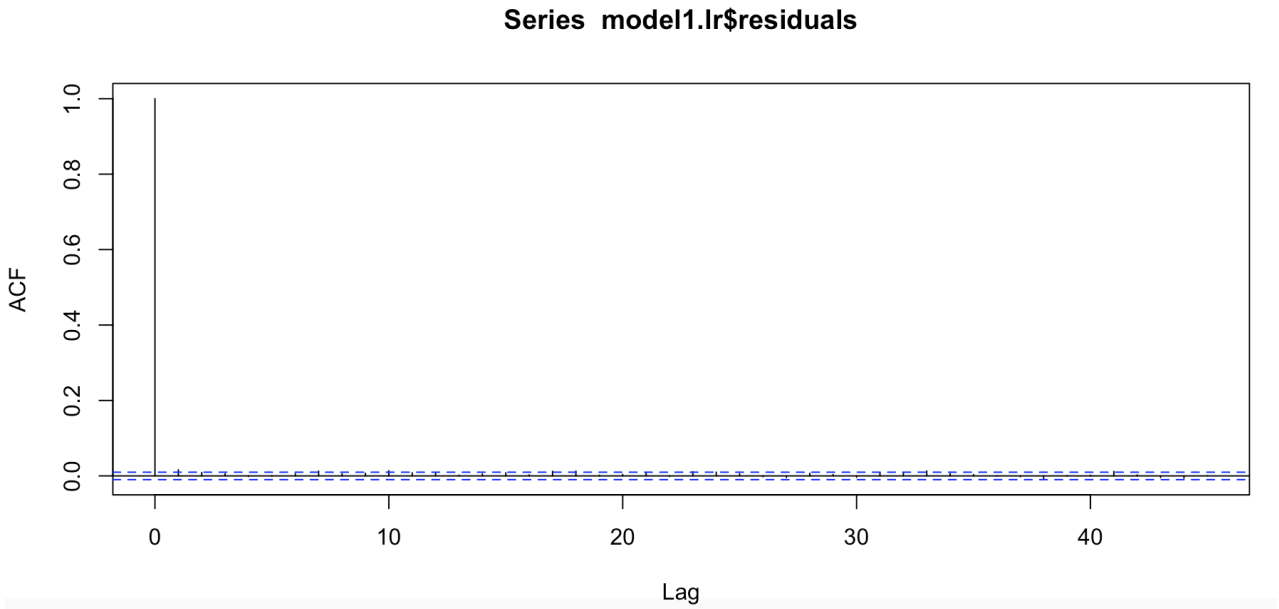

Figure 5 Correlogram for residuals of the linear model

It is evident from this analysis that linear regression to predict the exact number of shares does not work with a high degree of accuracy. It will be more fruitful to work on classes rather than trying to predict exact numbers.

4.2 Logistic Regression

Next I performed a binomial logistic regression on the *classes2* variable, which is the binary classification variable.

```
> summary(PredONP.logit)

Call:
glm(formula = classes2 ~ kw_avg_avg + self_reference_avg_sharess +
    LDA_03 + data_channel_is_world + num_imgs + global_subjectivity +
    num_hrefs + kw_min_avg + is_weekend + kw_min_max + kw_max_min +
    num_videos + data_channel_is_socmed + title_subjectivity +
    data_channel_is_entertainment + avg_positive_polarity + average_token_length +
    data_channel_is_lifestyle + LDA_01 + LDA_02 + data_channel_is_bus,
    family = binomial, data = TrainONP)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.2962  -1.0439  -0.6728   1.0737   1.9071

Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                  1.372e+00  2.429e-01   5.648 1.63e-08 ***
kw_avg_avg                    3.658e-01  2.002e-02  18.274  < 2e-16 ***
self_reference_avg_sharess     1.791e-01  1.322e-02  13.553  < 2e-16 ***
LDA_03                       -9.764e-01  6.542e-02 -14.926  < 2e-16 ***
data_channel_is_world1        -3.083e-01  5.819e-02  -5.299 1.17e-07 ***
num_imgs                     1.144e-02  2.502e-03   4.573 4.82e-06 ***
global_subjectivity           8.079e-01  1.749e-01   4.620 3.84e-06 ***
num_hrefs                    1.147e-02  1.835e-03   6.251 4.08e-10 ***
kw_min_avg                   -5.182e-05  1.760e-05  -2.945  0.00323 **
is_weekend1                   8.594e-01  3.849e-02  22.329  < 2e-16 ***
kw_min_max                   -4.065e-08  1.396e-06  -0.029  0.97677
kw_max_min                    1.124e-04  2.086e-05   5.389 7.10e-08 ***
num_videos                   2.286e-02  9.474e-03   2.414  0.01580 *
data_channel_is_socmed1       6.619e-01  6.043e-02  10.953  < 2e-16 ***
title_subjectivity           1.137e-01  3.895e-02   2.919  0.00352 **
data_channel_is_entertainment1 -4.791e-01  4.568e-02 -10.487  < 2e-16 ***
avg_positive_polarity        -7.453e-01  1.822e-01  -4.091 4.30e-05 ***
average_token_length         -2.366e-01  5.084e-02  -4.654 3.26e-06 ***
data_channel_is_lifestyle1    -2.867e-01  5.956e-02  -4.814 1.48e-06 ***
LDA_01                       -1.058e+00  8.750e-02 -12.088  < 2e-16 ***
LDA_02                       -9.191e-01  8.662e-02 -10.611  < 2e-16 ***
data_channel_is_bus1         -3.599e-01  4.079e-02  -8.822  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 41214  on 29732  degrees of freedom
Residual deviance: 37577  on 29711  degrees of freedom
AIC: 37621

Number of Fisher Scoring iterations: 4
```

Using the ROCR package, I calculated the area under the ROC curve and it is a decent 69.31% with accuracy of 64.71%

```
> print(paste("Accuracy for Logit=", round(sum(diag(confusionM.logit))/sum(confusionM.logit),4)*100, "%"))
[1] "Accuracy for Logit= 64.71 %"
> print(paste("AUC for Logit=", round(performance(ROCRpred, "auc")@y.values[[1]], digits = 4)*100, "%"))
[1] "AUC for Logit= 69.31 %"
```

The sensitivity and specificity calculated at a 0.494 threshold level were 65% and 64.4%. The threshold level in the figure below is calculated as the 'best' value based on 'youden' method, which maximized the distance from the identity line, which is the diagonal in the below figure.

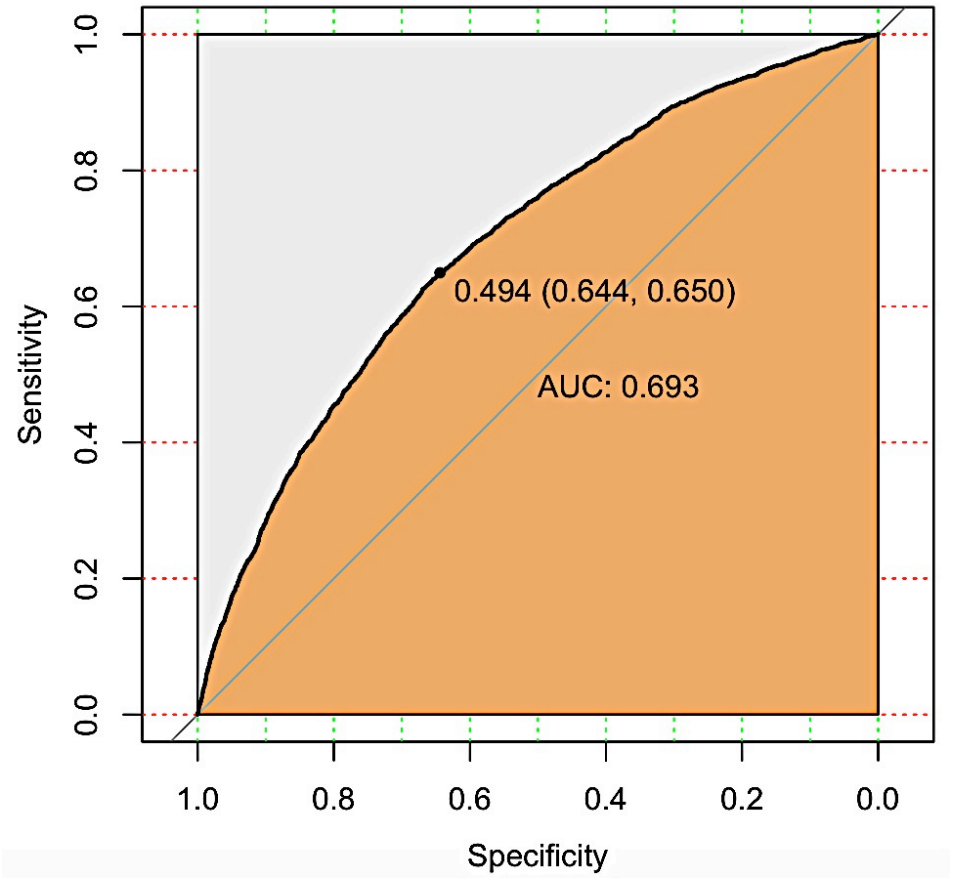

Figure 6 Sensivity vs. Specificity ROC Curve for the Logistic Regression

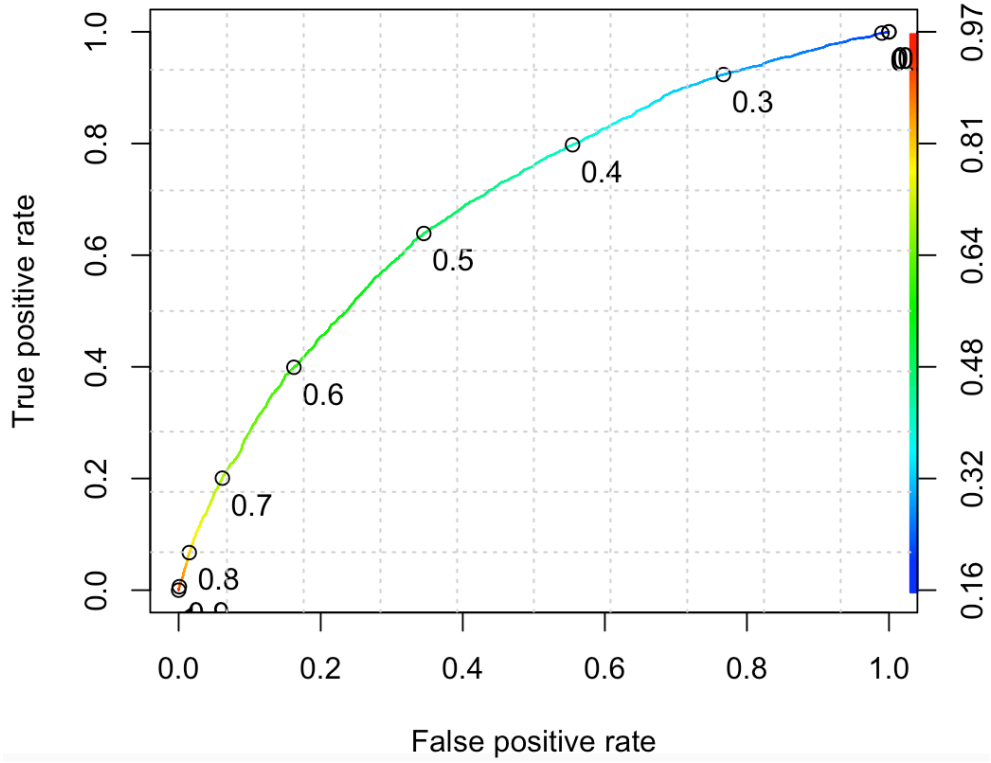

Figure 7 TPR vs. FPR ROC Curve for Logistic Regression

4.3 CART Binary Classification Using Cross-Validation

Following up logistic regression, I tried a CART machine-learning model. I used *classes2* as the dependent variable for learning. I used cross-validation of get the 'complexity parameter' for the CART model. Training the model using cross-validation returns a complexity parameter of 0.01.

```
CART

39644 samples
  18 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 4 times)
Summary of sample sizes: 35680, 35678, 35680, 35680, 35679, 35680, ...
Resampling results across tuning parameters:

 cp    Accuracy   Kappa
 0.01  0.6169405  0.23274214
 0.02  0.6153137  0.22934745
.
.
.
 0.49  0.5060707  0.00000000
 0.50  0.5060707  0.00000000
Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.01.
```

Next, I built the actual CART model using this parameter. The accuracy and AUC obtained from this model, at 61.6% and 62.4% respectively. This was lower than the logistic regression model.

```
> confusionMatrix(predictONP.class, TestONP$classes2)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3320 2091
         1 1715 2785

               Accuracy : 0.616
                 95% CI : (0.6063, 0.6256)
    No Information Rate : 0.508
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2308
 Mcnemar's Test P-Value : 1.213e-09

            Sensitivity : 0.6594
            Specificity : 0.5712
         Pos Pred Value : 0.6136
         Neg Pred Value : 0.6189
             Prevalence : 0.5080
         Detection Rate : 0.3350
   Detection Prevalence : 0.5460
      Balanced Accuracy : 0.6153
```

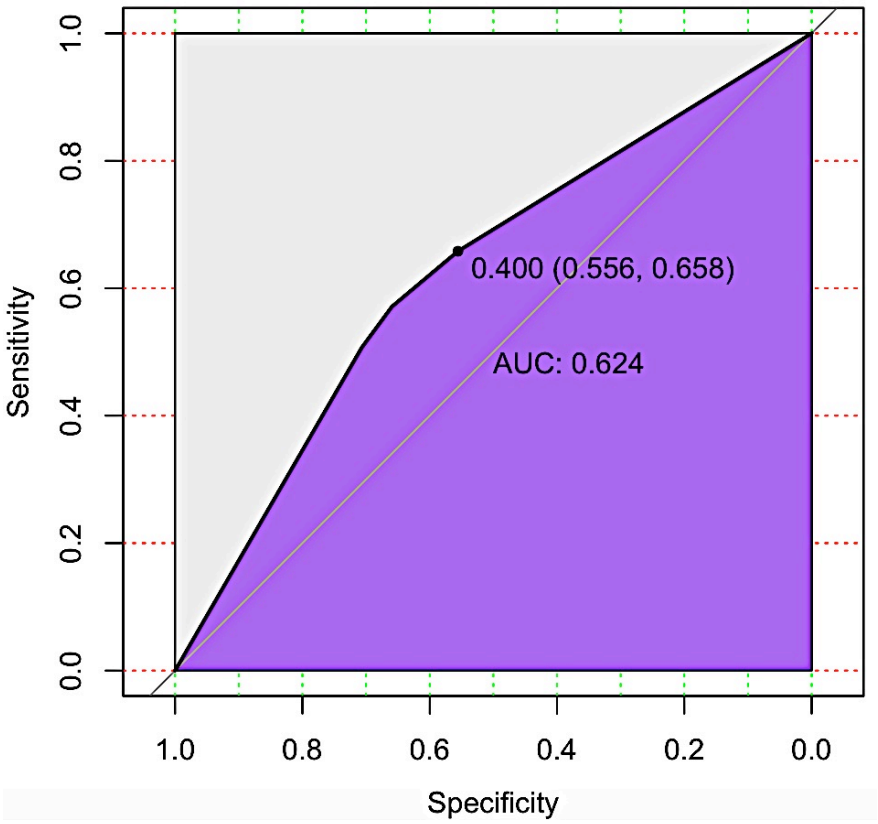

Figure 8 Sensitivity vs. Specificity ROC Curve for CART Model

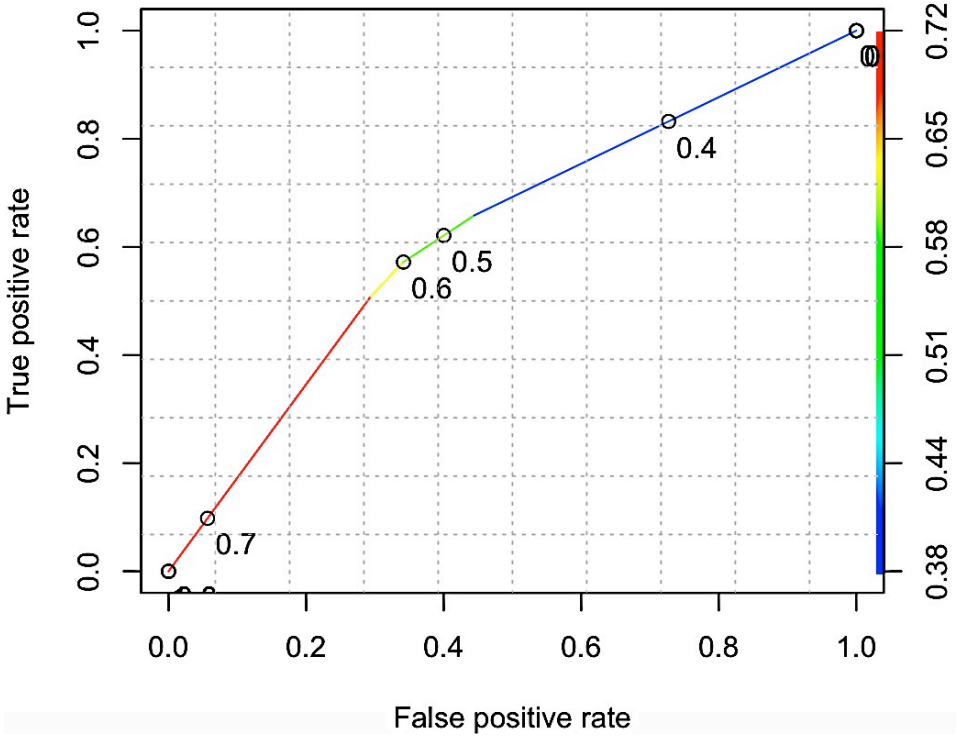

Figure 9 TPR vs. FPR ROC Curve for CART Model

The decision tree obtained from this CART model is shown below. This was obtained using the fancyRpartPlot() in the 'rattle' package.

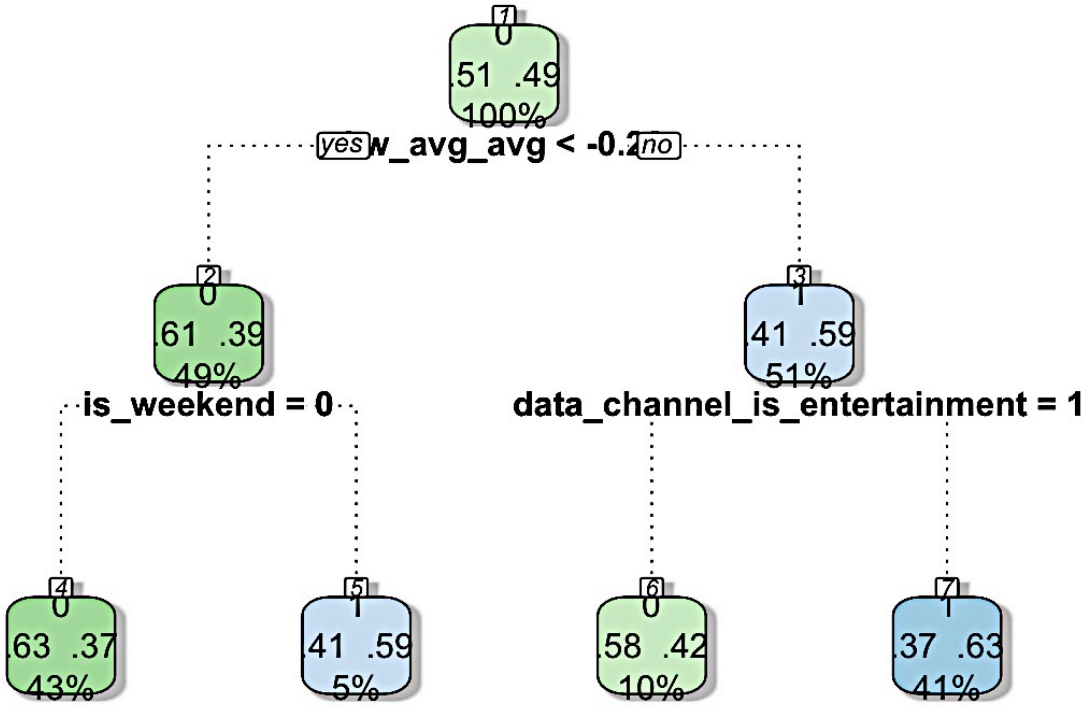

Figure 10 Decision tree used in CART Model

4.3 Random Forest Binary Classification

Since CART model returned results less accurate than logistic regression, I tried the Random Forest machine-learning model for classification. Again, I used *classes2* variable as the dependent variable in this classification. I tuned this model retroactively using the error plot with different values of ntree, nodesize, and mtry parameters. An error plot of the model shows that errors for both the classes go down and stabilize around 500 trees mark. The results from this model were superior to all the previous models.

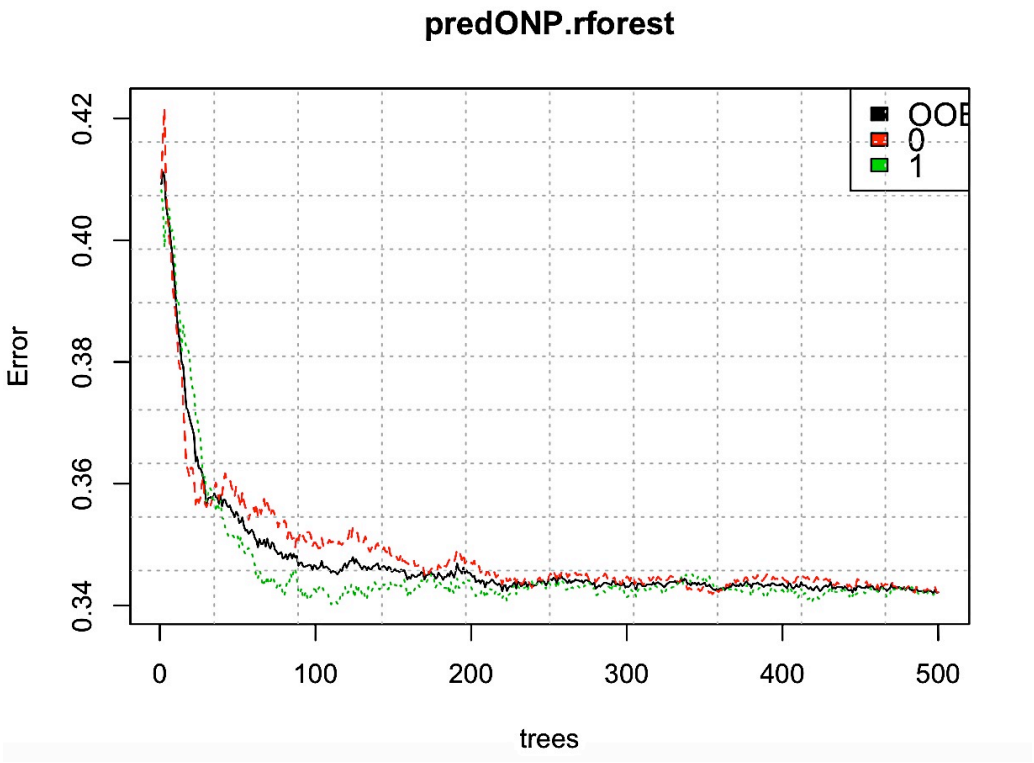

Figure 11 Error Plot for Classes using Binary Random Forest

```
> confusionMatrix(predictONP.forsest.2class, TestONP$classes2)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3238 1614
         1 1797 3262

               Accuracy : 0.6558
                 95% CI : (0.6464, 0.6652)
    No Information Rate : 0.508
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3119
 Mcnemar's Test P-Value : 0.001832

            Sensitivity : 0.6431
            Specificity : 0.6690
         Pos Pred Value : 0.6674
         Neg Pred Value : 0.6448
             Prevalence : 0.5080
         Detection Rate : 0.3267
   Detection Prevalence : 0.4896
      Balanced Accuracy : 0.6560
```

As we can see, accuracy for this model is 65.58% and AUC is 71.6% . The ROC curves below also show sensitivity and specificity of 71.7% and 60.1% respectively at a threshold of 0.467.

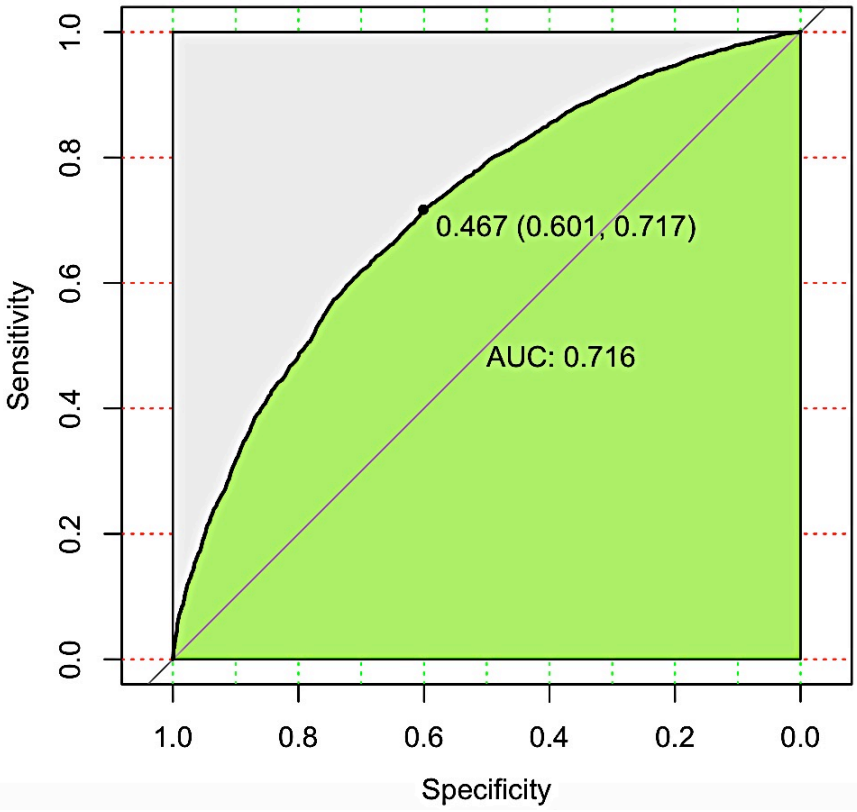

Figure 12 Sensivity vs. Specificity ROC Curve for Binary Random Forest

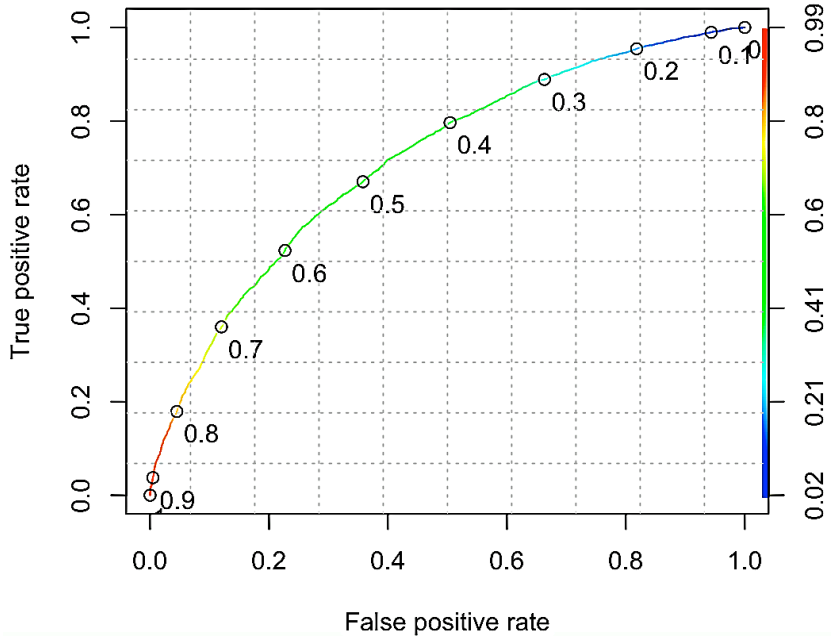

Figure 13 TPR vs. FPR ROC curve for Binary Random Forest

4.4 Random Forest 4-class Classification

Following the promising results from the previous Random Forest model, I tried to run this model on four classes instead of two. I used the *classes* variable as the dependent variable in this model, with 260 trees. As before, this was done retroactively using the error plot of classes. As we can see below, error for class 3 increases with increasing number of trees while for other classes the error actually decreases as the trees increase, along with the Out-of-Bag error.

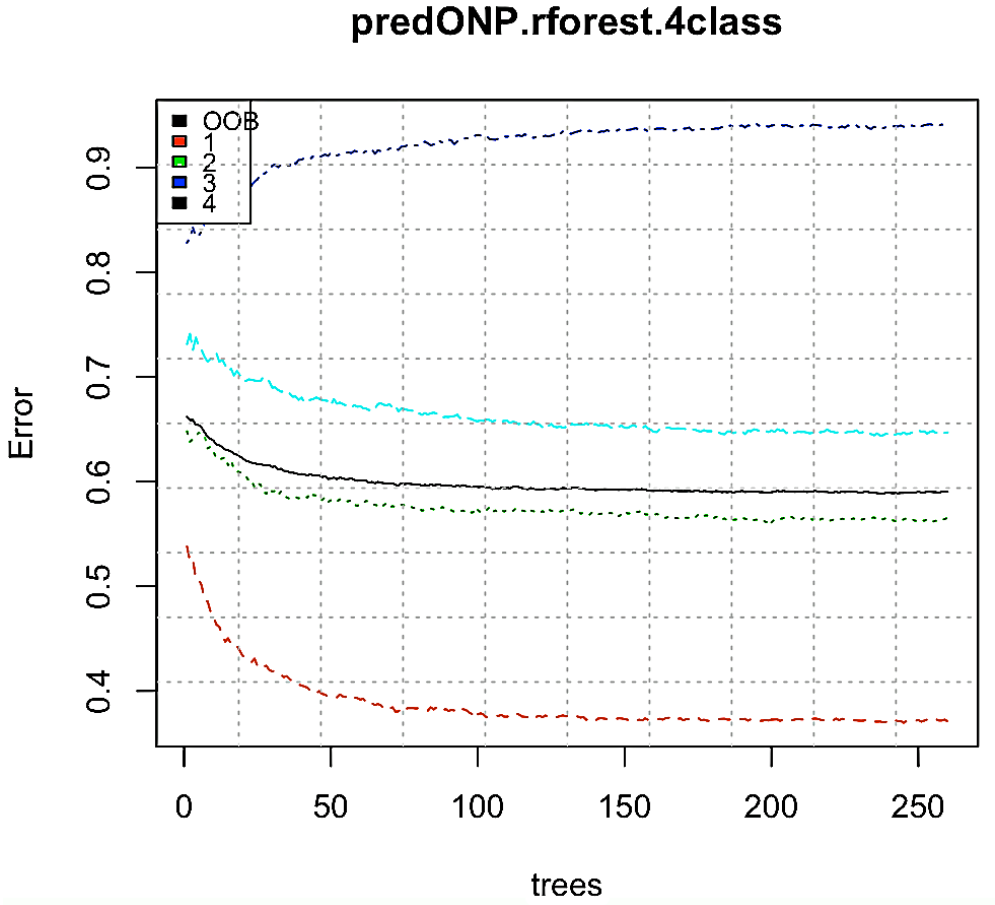

Figure 14 Error Plot for classes on 4-class Random Forest

Even with four classes, this model gives a decent accuracy of 41.3%, calculated using the Exact Match Ratio method, keeping in mind the fact that baseline for this model is 25%. The performance evaluation for this model was done using MAUC (Multi-Class AUC) method. MAUC calculates AUC for each class individually against all the other classes like with binary classes and then averages the individual AUCs for all the classes.

```
> Predicted.4class = PredictONP.forest.4class.prob/rowSums(PredictONP.forest.4class.prob)
> print(paste("Accuracy for 4-class RF =",
round(sum(diag(confusion.matrix))/sum(confusion.matrix),4)*100, "%"))
[1] "Accuracy for 4-class RF = 41.73 %"
> mauc(TestONP$classes, Predicted.4class)
$mauc
[1] 0.6604679

$auc
[1] 0.7253096 0.5746290 0.6252188 0.7167143

> ## Show that MAUC is the mean of individual AUCs
> mean(mauc(TestONP$classes, Predicted.4class)$auc)
[1] 0.6604679
```

5. Conclusion

It was evident from the low R squared on the linear model that trying to predict exact values was not a fruitful exercise. Predictions on the online news data are best handled as classification problems. It became clear during the course of my analysis why Mashable drew the line for binary classification at 1400 shares. It was the 50th quantile. I proceeded from there studying the data as a binary classification problem, and tried the models on a 4-class problem.

My conclusion from this study is that among all the techniques I used for the analysis of this problem, Random Forest by far works the best on both 2-class and 4-class classification. Using a rigorous feature selection process with finely tuned parameters for Random Forest, I was able to achieve an accuracy of 65.58 % and a decent accuracy of 41.3% on 4-class classification, which more realistically reflects the actual share categories of Mashable.com news articles.

References:

- He Ren, Quan Yang. "Predicting and Evaluating the Popularity of Online News". Link
- Johannes Ledolter, "Data Mining and Business Analytics with R"
- Rui Wang, Ke Tang, "An Empirical Study of MAUC in Multi-class Problems with Uncertain Cost Matrices". Link
- R-package library at CRAN (https://cran.r-project.org/web/packages/) and other places.
- Springboard student resources