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 39,797 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

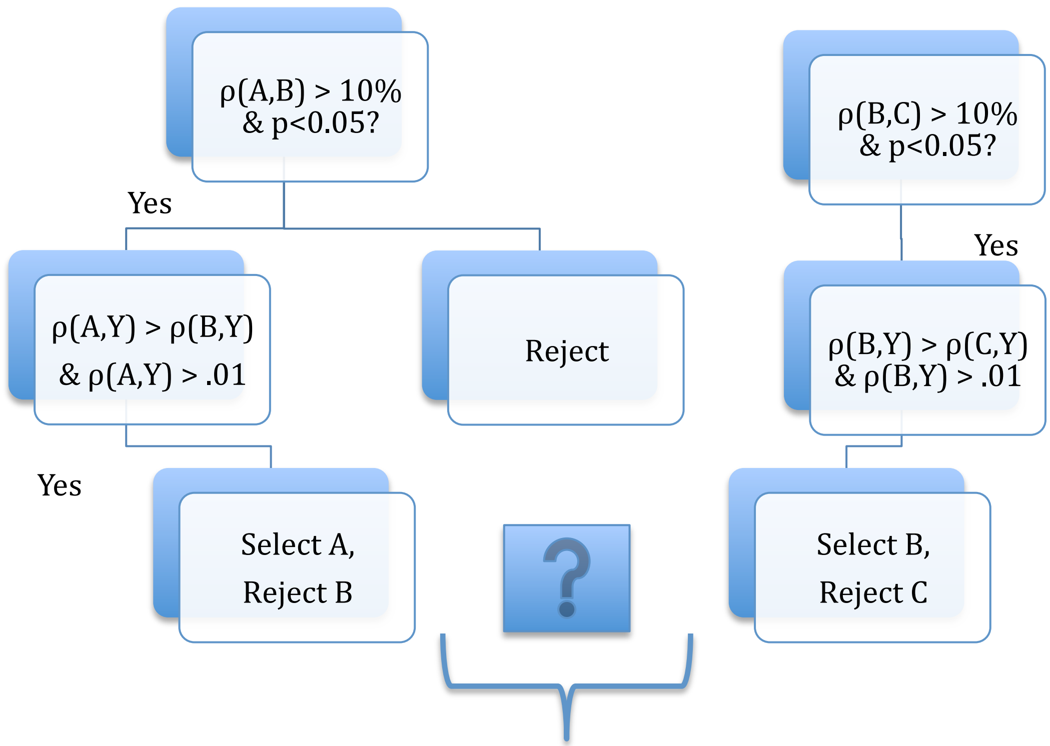
We need to select only the significant features of the model to reduce complexity and multicollinearity in the models. Principal Component Analysis (PCA) or Multi Factor Analysis (MFA) did not seem particularly suitable as feature selection techniques as my variable set contains a mixture of binary factors 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).

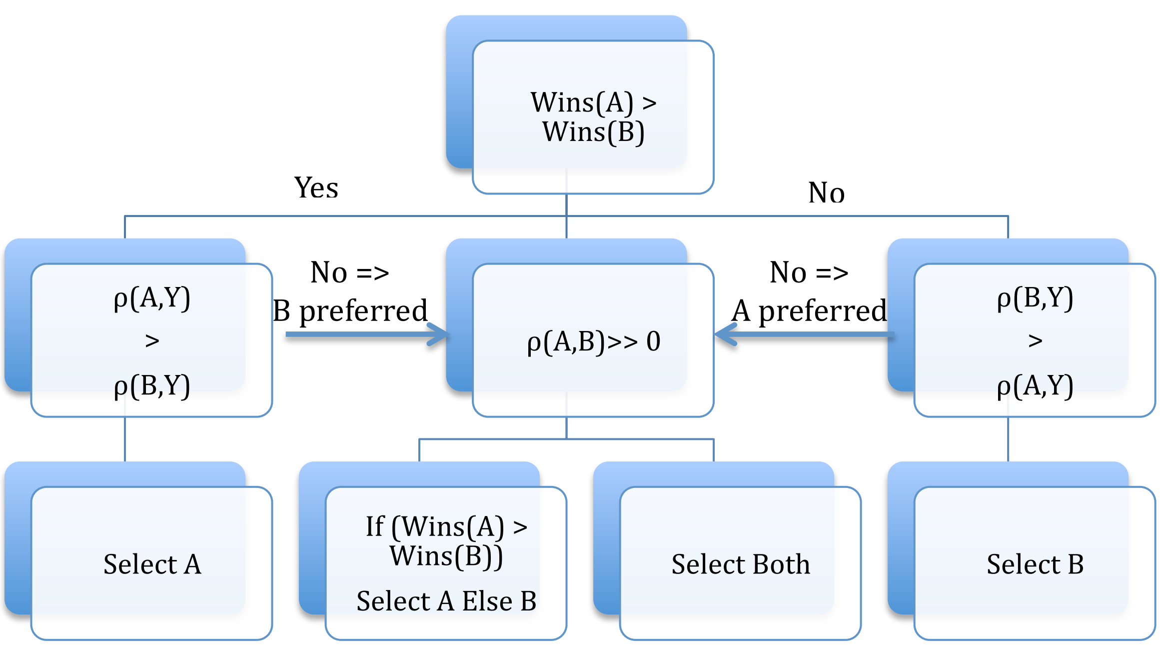
The analysis described above does not yield clean solutions. This is because the above selection process has to deal with contests between pairs of variables that are not mutually exclusive. To give an example, *avg_positive_polarity* is a choice variable between many comparisons, but between avg_positive_polarity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


ρ(A,B) > 10% & p<0.05?
ρ(B,C) > 10% & p<0.05?
Yes
Yes
ρ(A,Y) > ρ(B,Y) & ρ(A,Y) > .01
Reject
ρ(B,Y) > ρ(C,Y) & ρ(B,Y) > .01
Yes
Select A, Reject B
Select B, Reject C


Wins(A) > Wins(B)
Yes
No
ρ(A,Y) > ρ(B,Y)
No => B preferred
ρ(A,B)>> 0
No => A preferred
ρ(B,Y) > ρ(A,Y)
Select A
If (Wins(A) > Wins(B)) Select A Else B
Select Both
Select B

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.

row	column	cor	p	selection	rcor	ccor
avg_positive_polarity	max_positive_polarity	0.70355815	0	avg_positive_polarity	0.0121422	0.01006779
avg_positive_polarity	min_positive_polarity	0.45697269	0	avg_positive_polarity	0.0121422	-4.01E-05
global_rate_negative_words	avg_positive_polarity	0.19298527	0	avg_positive_polarity	0.00661517	0.0121422
global_rate_positive_words	avg_positive_polarity	0.33132637	0	avg_positive_polarity	0.00054323	0.0121422
global_sentiment_polarity	avg_positive_polarity	0.49670392	0	avg_positive_polarity	0.00416293	0.0121422
n_tokens_content	avg_positive_polarity	0.13512254	0	avg_positive_polarity	0.00245898	0.0121422
rate_negative_words	avg_positive_polarity	0.14568889	0	avg_positive_polarity	-0.0051831	0.0121422
timedelta	avg_positive_polarity	0.12634392	0	avg_positive_polarity	0.00866229	0.0121422
abs_title_subjectivity	abs_title_sentiment_polarity	-0.4002718	0	abs_title_sentiment_polarity	0.001481	0.02713523
avg_positive_polarity	abs_title_sentiment_polarity	0.10119107	0	abs_title_sentiment_polarity	0.0121422	0.02713523
global_rate_positive_words	abs_title_sentiment_polarity	0.10339063	0	abs_title_sentiment_polarity	0.00054323	0.02713523
title_sentiment_polarity	abs_title_sentiment_polarity	0.41020525	0	abs_title_sentiment_polarity	0.01277187	0.02713523
title_subjectivity	abs_title_sentiment_polarity	0.71452761	0	abs_title_sentiment_polarity	0.02196668	0.02713523

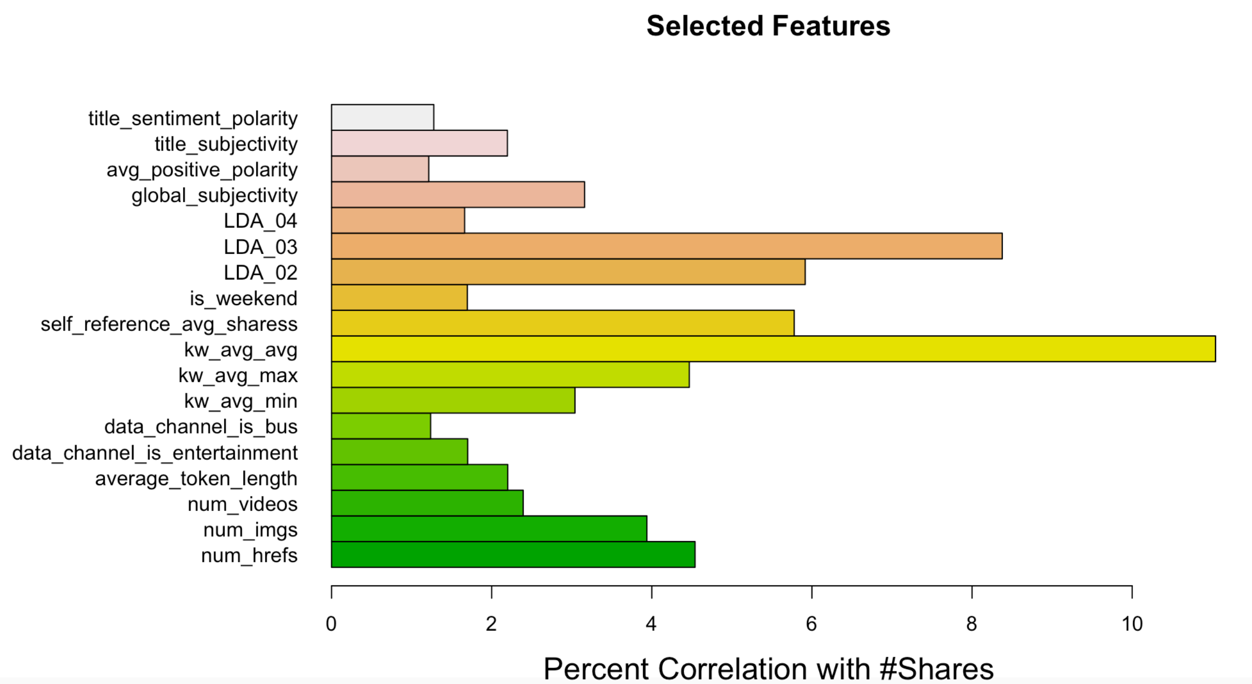
According to the scheme defined above, we observe that between avg_positive_polarity and abs_title_sentiment_polarity, avg_positive_polarity has the double the number of wins but abs_title_sentiment_polarity has double the correlation with number of shares. So there's no clear choice. We then see whether 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 21 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 18.

The following features made the final cut:

1	num_hrefs
2	num_imgs
3	num_videos
4	average_token_length
5	data_channel_is_entertainment
6	data_channel_is_bus
7	kw_avg_min
8	kw_avg_max
9	kw_avg_avg
10	self_reference_avg_sharess
11	is_weekend
12	LDA_02
13	LDA_03
14	LDA_04

15	global_subjectivity
16	avg_positive_polarity
17	title_subjectivity
18	title_sentiment_polarity



3. Data Preparation

3.1 Capping
Once the significant features are obtained, 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 except the first one which contains the actual news link.

3.2 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. shares
2. self_reference_min_shares
3. self_reference_avg_sharess
4. self_reference_max_shares
5. kw_avg_min
6. kw_avg_max
7. kw_avg_avg

A new variable "scaledy" is introduced above, which is nothing but scaled 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.

3.3 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))
   10%     20%     30%     40%     50%     60%     70%     80%     90%    100%
 708.0   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
```

3.4 Factoring

To be able to tell my regression or classification models that the above two variables are indeed classes, I converted them to factors. Moreover, there are three dependent binary variables among the one that we chose that need to be factorized:

1. data_channel_is_bus
2. data_channel_is_entertainment
3. is_weekend

3.5 Splitting

I split the data into training and test sets on a 70%-30% data split.

4. Models and Results

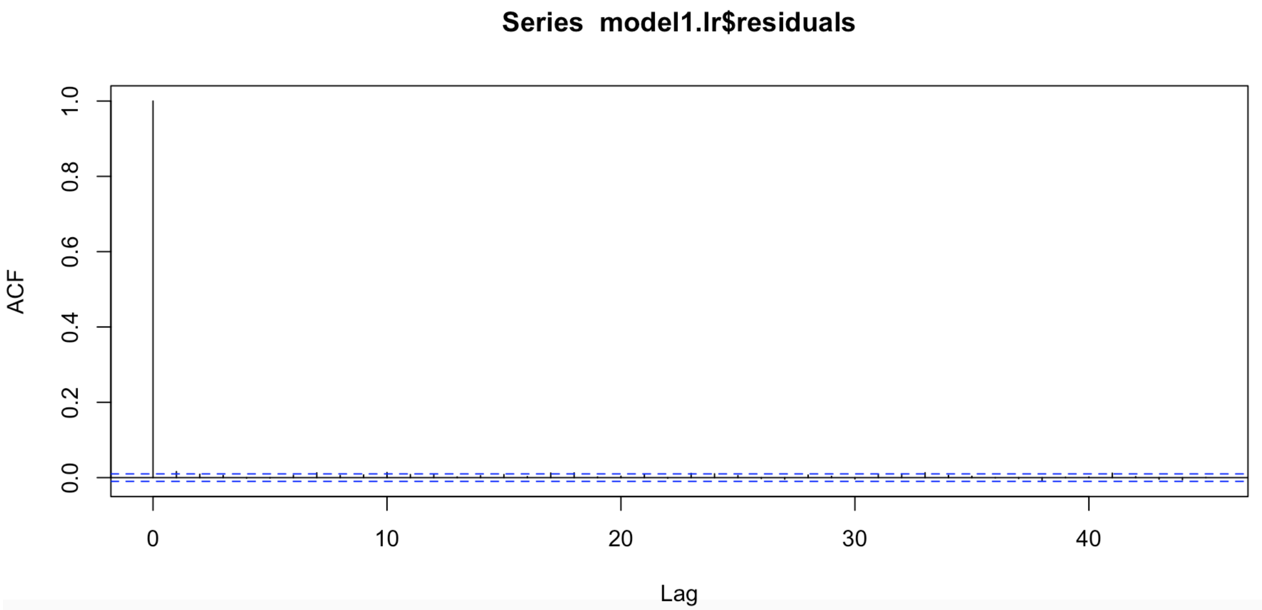
4.1 Linear Regression

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 0.09.

```
> summary(model1.lr)

Call:
lm(formula = scaledy ~ average_token_length + avg_positive_polarity +
    factor(data_channel_is_bus) + factor(data_channel_is_entertainment) +
    global_subjectivity + factor(is_weekend) + kw_avg_avg + kw_avg_max +
    kw_avg_min + LDA_02 + LDA_03 + LDA_04 + num_hrefs + num_imgs +
    num_videos + self_reference_avg_sharess + title_sentiment_polarity +
    title_subjectivity, data = TrainONP)

Residuals:
    Min      1Q  Median      3Q     Max
-1.7604 -0.5126 -0.2803  0.0913  3.6871

Coefficients:
                                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                            0.5785775  0.0956935   6.046 1.50e-09 ***
average_token_length                  -0.1253167  0.0196303  -6.384 1.75e-10 ***
avg_positive_polarity                 -0.3010560  0.0705611  -4.267 1.99e-05 ***
factor(data_channel_is_bus)1          -0.1122887  0.0187793  -5.979 2.26e-09 ***
factor(data_channel_is_entertainment)1 -0.2475756  0.0156570 -15.812  < 2e-16 ***
global_subjectivity                    0.3510964  0.0676835   5.187 2.14e-07 ***
factor(is_weekend)1                    0.1754585  0.0142610  12.303  < 2e-16 ***
kw_avg_avg                             0.2024007  0.0073263  27.627  < 2e-16 ***
kw_avg_max                            -0.0649528  0.0072084  -9.011  < 2e-16 ***
kw_avg_min                             0.0120977  0.0058217   2.078 0.037713 *
LDA_02                                -0.2846464  0.0272197 -10.457  < 2e-16 ***
LDA_03                                -0.0658378  0.0254020  -2.592 0.009550 **
LDA_04                                -0.0876990  0.0253116  -3.465 0.000531 ***
num_hrefs                              0.0045111  0.0007031   6.416 1.42e-10 ***
num_imgs                               0.0043067  0.0009602   4.485 7.30e-06 ***
num_videos                             0.0156842  0.0036810   4.261 2.04e-05 ***
self_reference_avg_sharess             0.1136776  0.0050017  22.728  < 2e-16 ***
title_sentiment_polarity               0.0738561  0.0247527   2.984 0.002849 **
title_subjectivity                     0.0408377  0.0158298   2.580 0.009889 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9495 on 39625 degrees of freedom
Multiple R-squared:  0.09894,	Adjusted R-squared:  0.09853
```

F-statistic: 241.7 on 18 and 39625 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.



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 ~ average_token_length + avg_positive_polarity +
   factor(data_channel_is_bus) + factor(data_channel_is_entertainment) +
   global_subjectivity + factor(is_weekend) + kw_avg_avg + kw_avg_max +
   kw_avg_min + LDA_02 + LDA_03 + LDA_04 + num_hrefs + num_imgs +
   num_videos + self_reference_avg_sharess + title_sentiment_polarity +
   title_subjectivity, family = binomial, data = TrainONP)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-2.3258  -1.0509  -0.6618   1.0909   2.0412

Coefficients:
```

```
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                        1.286594   0.213200   6.035 1.59e-09 ***
average_token_length               -0.250468   0.043810  -5.717 1.08e-08 ***
avg_positive_polarity              -0.926314   0.157781  -5.871 4.34e-09 ***
factor(data_channel_is_bus)1         -0.163933   0.041469  -3.953 7.71e-05 ***
factor(data_channel_is_entertainment)1 -0.779142   0.035168 -22.155  < 2e-16 ***
global_subjectivity                0.799267   0.151011   5.293 1.20e-07 ***
factor(is_weekend)1                0.814059   0.033249  24.483  < 2e-16 ***
kw_avg_avg                         0.400066   0.016609  24.087  < 2e-16 ***
kw_avg_max                        -0.156352   0.016050  -9.742  < 2e-16 ***
kw_avg_min                         0.071447   0.013023   5.486 4.11e-08 ***
LDA_02                            -0.972450   0.060903 -15.967  < 2e-16 ***
LDA_03                            -0.539889   0.057322  -9.419  < 2e-16 ***
LDA_04                             0.039890   0.056310   0.708 0.478695
num_hrefs                          0.012721   0.001583   8.035 9.36e-16 ***
num_imgs                           0.007705   0.002153   3.579 0.000346 ***
num_videos                         0.035740   0.008281   4.316 1.59e-05 ***
self_reference_avg_sharess         0.205558   0.011450  17.952  < 2e-16 ***
title_sentiment_polarity           0.307797   0.055531   5.543 2.98e-08 ***
title_subjectivity                 0.017524   0.035420   0.495 0.620786
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 54951  on 39643  degrees of freedom
Residual deviance: 50387  on 39625  degrees of freedom
AIC: 50425

Number of Fisher Scoring iterations: 4
```

Using the ROCR package, I calculated the area under the ROC curve and it is a decent 69.42%.

```
> print(paste("AUC =", round(performance(ROCRpred, "auc")@y.values[[1]], digits = 4)*100, "%"))
[1] "AUC = 69.42 %"
```

The accuracy for this model is also a decent 64.38%

```
> confusionM.logit = table(onp$classes2, predictONP.logit > 0.5)
> print(paste("Accuracy =", round(sum(diag(confusionM.logit))/sum(confusionM.logit),4)*100, "%"))
[1] "Accuracy = 64.38 %"
```

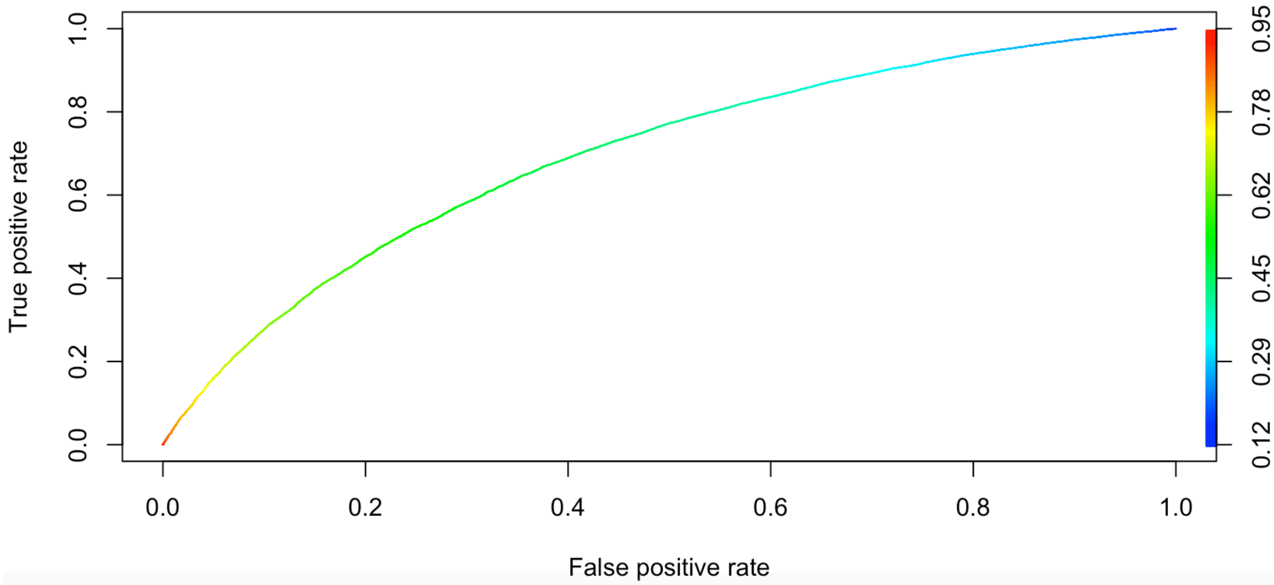
4.3 CART Binary Classification Using Cross-Validation

Following up logistic regression, I try 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
.
.
.
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 was lower than the logistic regression model.

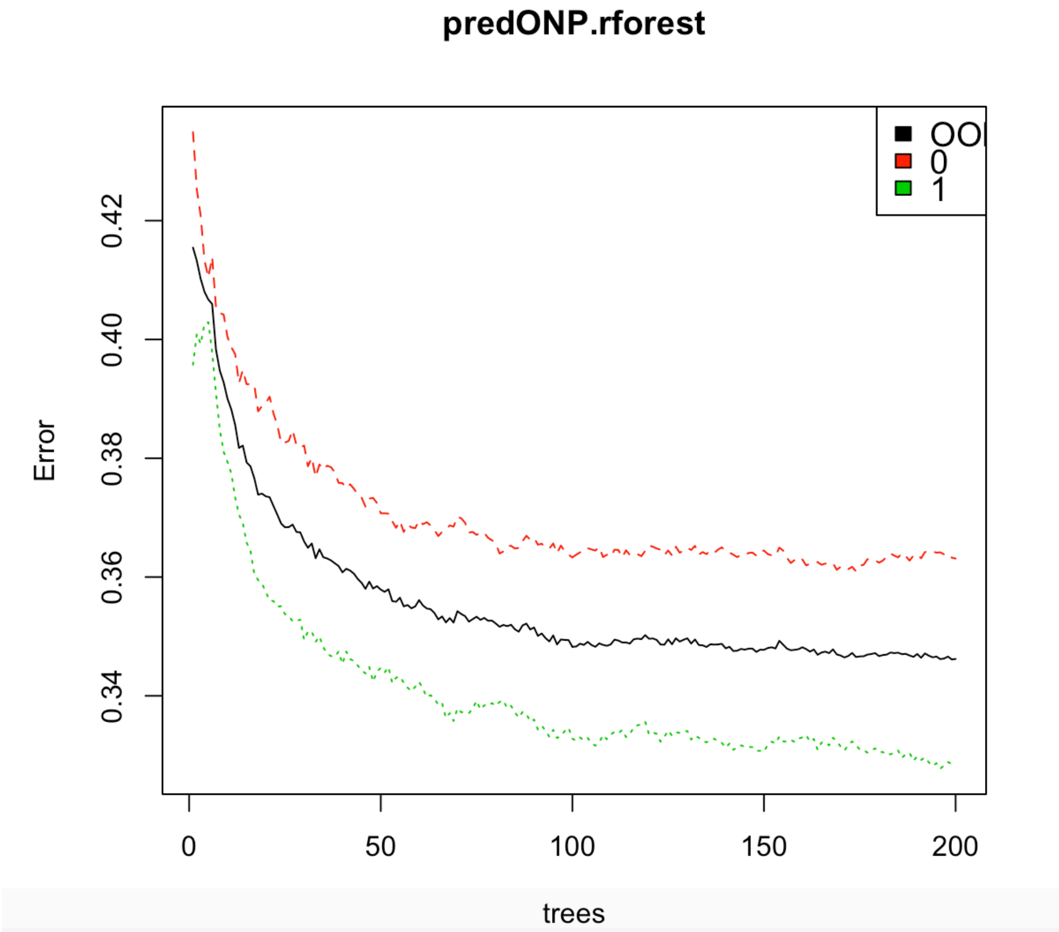
```
> confusionM.cart = table(TestONP$classes2, predictONP.class)
> print(paste("Accuracy =", round(sum(diag(confusionM.cart))/sum(confusionM.cart),4)*100, "%"))
[1] "Accuracy = 61.94 %"
> print(paste("AUC =", round(performance(pred.class, "auc")@y.values[[1]], digits = 4)*100, "%"))
[1] "AUC = 62.53 %"
```



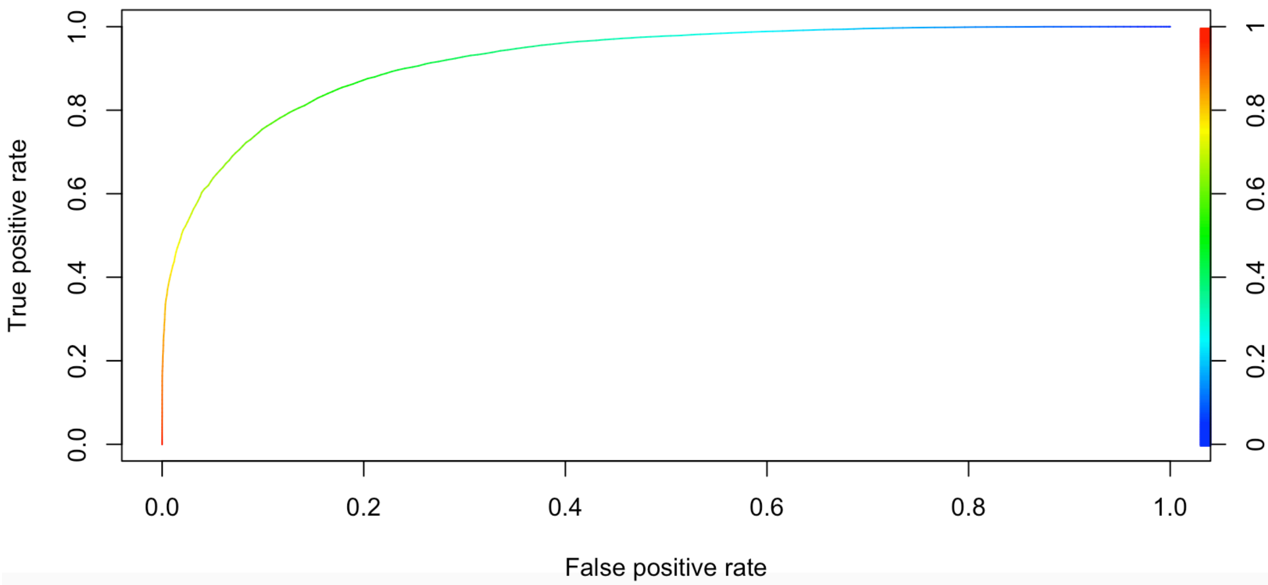
As we can see, accuracy is 61.94% and AUC is 62.53%.

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 used 200 trees for this model with node size of 25. An error plot of the model shows that errors or both classes go down and stabilized around 200 trees mark. The results from this model were superior to all the previous models.



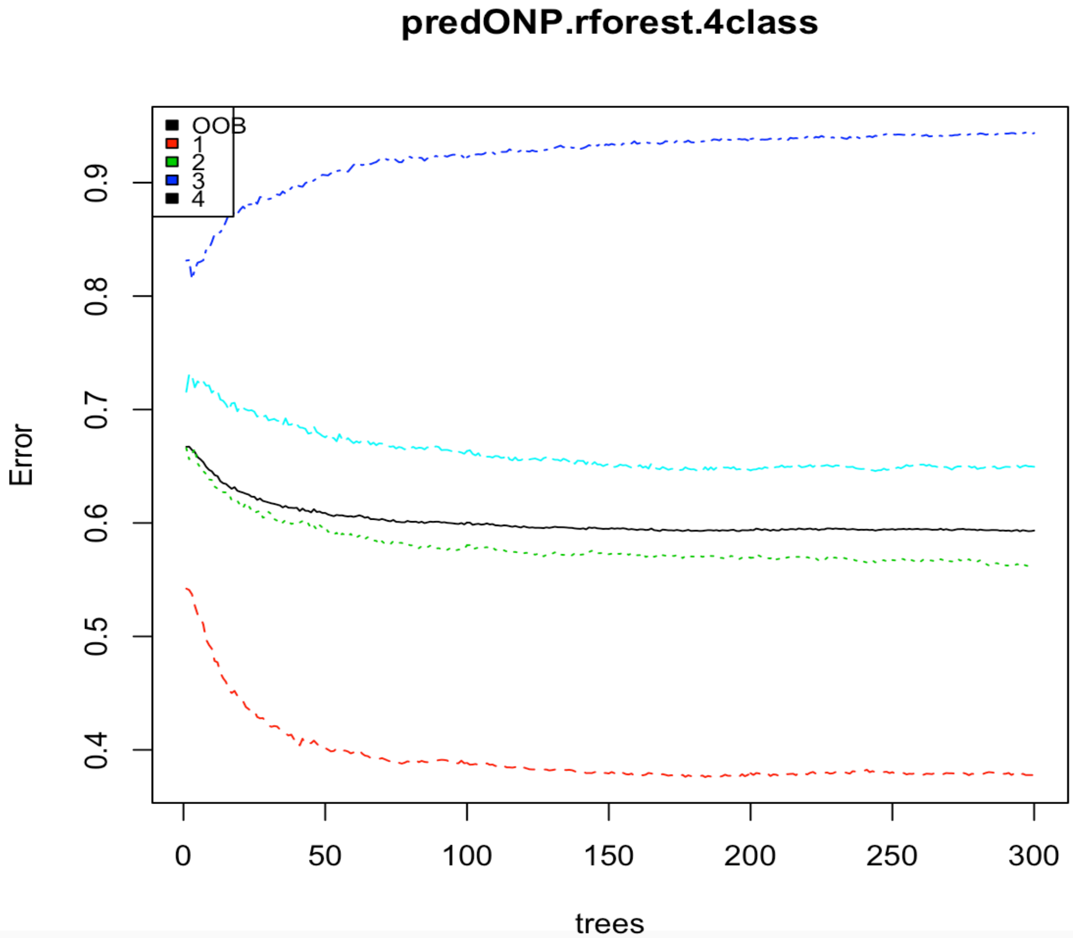

```
> confusionM.RF = table(TestONP$classes2, predictONP.forsest.2class)
> print(paste("Accuracy =", round(sum(diag(confusionM.RF))/sum(confusionM.RF),4)*100, "%"))
[1] "Accuracy = 83.75 %"
> print(paste("AUC =", round(performance(predrocr, "auc")@y.values[[1]], digits = 4)*100, "%"))
[1] "AUC = 92.19 %"
```



As we can see, accuracy for this model is 83.75% and AUC is 92.19%.

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 300 trees. As we can see below, error for class 3 increases with increasing number of trees while for other classes and the error actually decreaseses as the trees increase, along with the Out-of-Bag error.



This Random forest model, as the previous one, makes predictions with high accuracy.

```
> print(paste("Accuracy =", round(sum(diag(confusion.matrix))/sum(confusion.matrix),4)*100, "%"))
[1] "Accuracy = 68.7 %"
```

Even with four classes, this model gives a decent accuracy of 68.7%, calculated using the Exact Match Ratio method.

5. Conclusion

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 83.75 %, higher than any other proposed solution currently available on the web. I was also able to achieve a decent accuracy of 68.7% on 4-class classification, which more realistically reflects the actual share categories of Mashable.com news articles.