Predicting Popularity of Online News Articles

Group*:*

Drake Carmichael
Michael Schniepp
Jerrick Zhang
Sklar Jones

Description:

More and more people are receiving news online these days and with the increasing use of the Internet there is an increasing amount of information to analyze. The dataset to be analyzed was composed of numeric data that described 59 different text metrics pertaining to 39644 different articles from the online news site Mashable.com. For example, some of these metrics are things such as number of links, number of words in the text, number of words in the title, subject category, and day the article was published (see appendix for whole attribute listing).

What we wanted to find out was, using these text-metrics, can we predict the popularity of an article? To measure popularity we used the number of shares an article has where a higher number of shares indicate a more popular article.

Methods of Analysis:

Exploratory Analysis
Due to the high dimensionality of the data, we attempted a PCA analysis to try to get a visualization of the data. The PCA visualization showed no clearly defined groupings leading us to believe that LDA or Cluster analysis would not offer much benefit to us.

Also due to the sheer size of the dimension of the data we felt regression techniques would fail to really capture the phenomenon of what makes an article popular. Thus we decided a classification method would be best. To do this we would attempt to determine if an article was either popular or unpopular and separated the data by the median.

KNN
Due to the computational limitation of our machines we were only able to test for a few values of k for the KNN algorithm. Through these few tests we found 5 to be optimal. As expected the KNN algorithm was not very effective in this classification problem due to the curse of dimensionality phenomenon. With a dimension of 59 measuring distances become a bit convoluted and arbitrary and the classification becomes quite inaccurate. Our results with KNN showed it was not much better than classifying at random.

Classification Tree
To perform a classification the first obvious choice was to try making a model with a standard classification tree. The classification tree did a mediocre job only yielding a 62% accuracy rate. Using some more refined methods could allow us to answer our question with much better precision.

Classification Trees with Bagging

To increase the accuracy of our tree model we implemented a Bagged tree (bootstrap aggregated). By resampling new data sets we can increase the variability of different trees that are generated supplying us with an even more accurate model. The bagged tree yielded a 64.0% accuracy rate, which is an improvement but still limits our predictive ability.

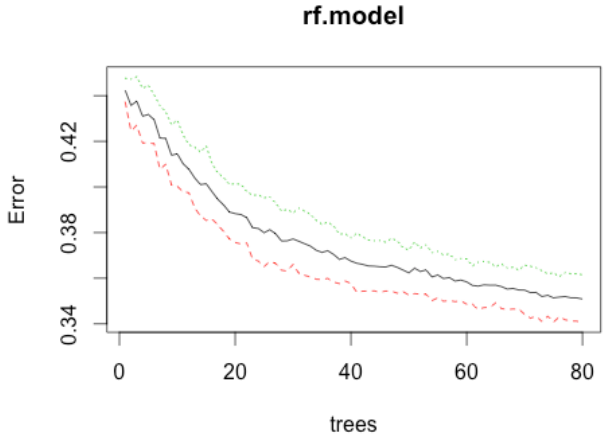
Random Forest

Next, we tried the Random Forest method. This included bagging the models as well as taking random samples of the predictors to make many different trees and then averaging the results. With this method we obtained our highest accuracy of 65.1%.

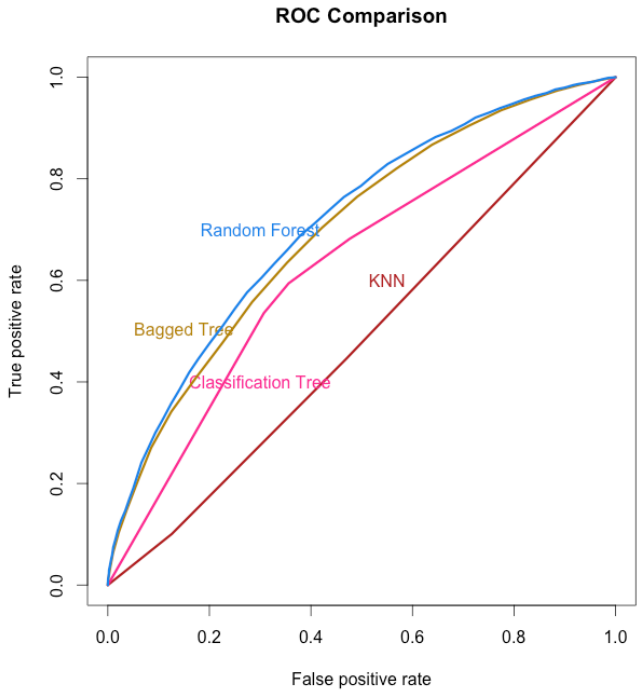
Method	Accuracy	AUC
Tree	61.9%	0.627
Bagged	64.0%	0.697
RF	65.1%	0.713
KNN	56.5%	0.483

Final Model Selection

We decided to use the Random Forest method for our final analysis because it gave us the highest prediction accuracy of all the methods we tried. If you take a look at the associated ROC comparison plot you can also see the Random Forest model was our best performing model, along with the highest Area Under the Curve (AUC) as shown in the above chart.



The rf.model plot for the random forest error showed that the error rate starts to level out around 80 trees generated. A similar result was shown for 100 trees but to save on computational costs we cut it down to 80 with minimal loss in accuracy.



Summary of Results

With the implementation of top-of-the-line algorithms the best accuracy percentage we could attain was 65.1%, which leads us to believe there is little room for improvement in model selection but rather feature selection.

The random forest algorithm is a very well refined and effective method for classifications and thus we would consider taking a look at ways to produce better results by using more indicative predictors.

Conclusion

We have found that this data set is not ideal to analyze and predict news popularity, as it begs for more information, mostly because the data is a result of superficial text mining results. They are very general and nearly arbitrary when it comes to predicting how popular an article may become. People tend to share different news at different times according to a multitude of global affairs happening at any given time, so without variables measuring the media environment and general current event conditions, it is hard to believe there is a general rule that can lead to the accurate prediction of an articles popularity.

PSTAT 131 Project

May 31, 2016

This is the R Code used for the data analysis for our project.

Reading in the Data and Creating working Datasets

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.2.3
```

```
# Data read-in
webData <- read.table("/Users/michaelschniepp/Desktop/131/Project/OnlineNewsPopularity/OnlineNewsPopula
                      header = TRUE, sep=",")
# Delete url and timedelta (non-predictive)
webData <- subset(webData, select = -c(url, timedelta ) )


# Working Datasets
webData.class <- webData
webData.class$shares <- as.factor(ifelse(webData.class$shares > 1400,1,0)) # median is 1400

# Set training and test sets
set.seed(1234)
index <- sample(nrow(webData.class), 27751, replace=FALSE)
trainSet <- webData.class[index,] # 70%
testSet <- webData.class[-index,] # 30%
```

Creating the Models

```
# CLASSIFICATION TREE MODEL

library(rpart)
classmodel.tree <- rpart(shares ~., data=trainSet, method="class")
classmodel.tree.pred <- predict(classmodel.tree, testSet ,type="class")
classmodel.tree.prob <- predict(classmodel.tree, testSet ,type="prob")

1 - mean(classmodel.tree.pred != testSet$shares) # Accuracy rate
```

```
## [1] 0.6191878
```

```
# BAGGED TREE MODEL

library(ipred)
bag.model <- bagging(shares ~., data=trainSet, coob=TRUE)
```

```
bag.model.pred <- predict(bag.model,testSet, type="class")
bag.model.prob <- predict(bag.model,testSet, type="prob")
1 - mean(bag.model.pred != testSet$shares) # accuracy
```

```
## [1] 0.6402926
```

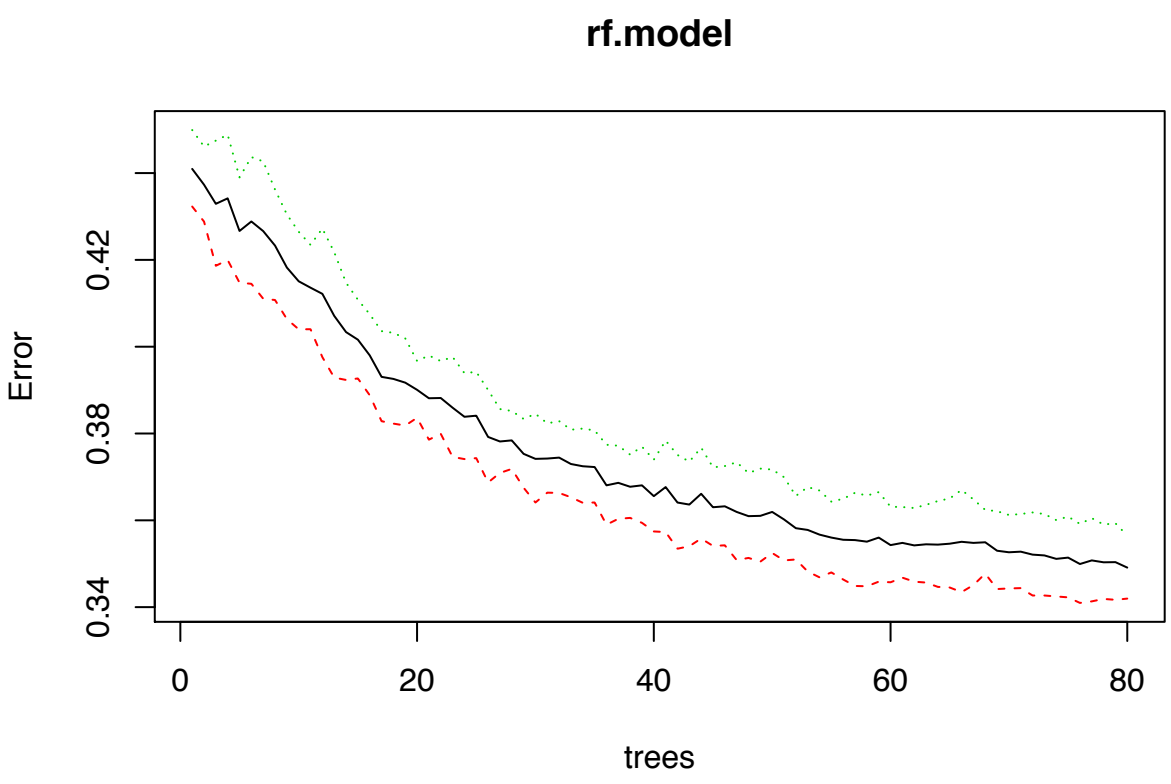
```
# RANDOM FOREST MODEL

library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.3
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
rf.model <- randomForest(trainSet[,1:58],trainSet[,59],
                         ntree=80,nPerm=10,mtry=3,proximity=FALSE,importance=TRUE)
plot(rf.model)
```




```
rf.model.pred <- predict(rf.model,testSet, type="class")
rf.model.prob <- predict(rf.model,testSet, type="prob")
1 - mean(rf.model.pred != testSet$shares) # accuracy
```

```
## [1] 0.6545867
```

```
# KNN MODEL
library(class)
webData.knn <- knn(trainSet[,-59], testSet[,-59], trainSet[,59], k = 5, prob = FALSE)
webData.knn.prob <- knn(trainSet[,-59], testSet[,-59], trainSet[,59], k = 5, prob = TRUE)
1 - mean(webData.knn != testSet$shares)
```

```
## [1] 0.5650383
```

Generating the Plots

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
library(ROCR)
```

```
## Loading required package: gplots
```

```
## Warning: package 'gplots' was built under R version 3.2.4
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 3.2.3
```

```
## Loading required package: proto
```

```
# ROC Curves
par(mfrow = c(2, 2))
classtree.roc <- roc(testSet$shares, classmodel.tree.prob[,2])
plot(classtree.roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("red", "red"), max.auc.polygon=TRUE,
     auc.polygon.col='seagreen1', print.thres=TRUE, main="Classification Tree")
```

```
##
## Call:
## roc.default(response = testSet$shares, predictor = classmodel.tree.prob[,     2])
##
## Data: classmodel.tree.prob[, 2] in 5990 controls (testSet$shares 0) < 5903 cases (testSet$shares 1).
## Area under the curve: 0.6269
```

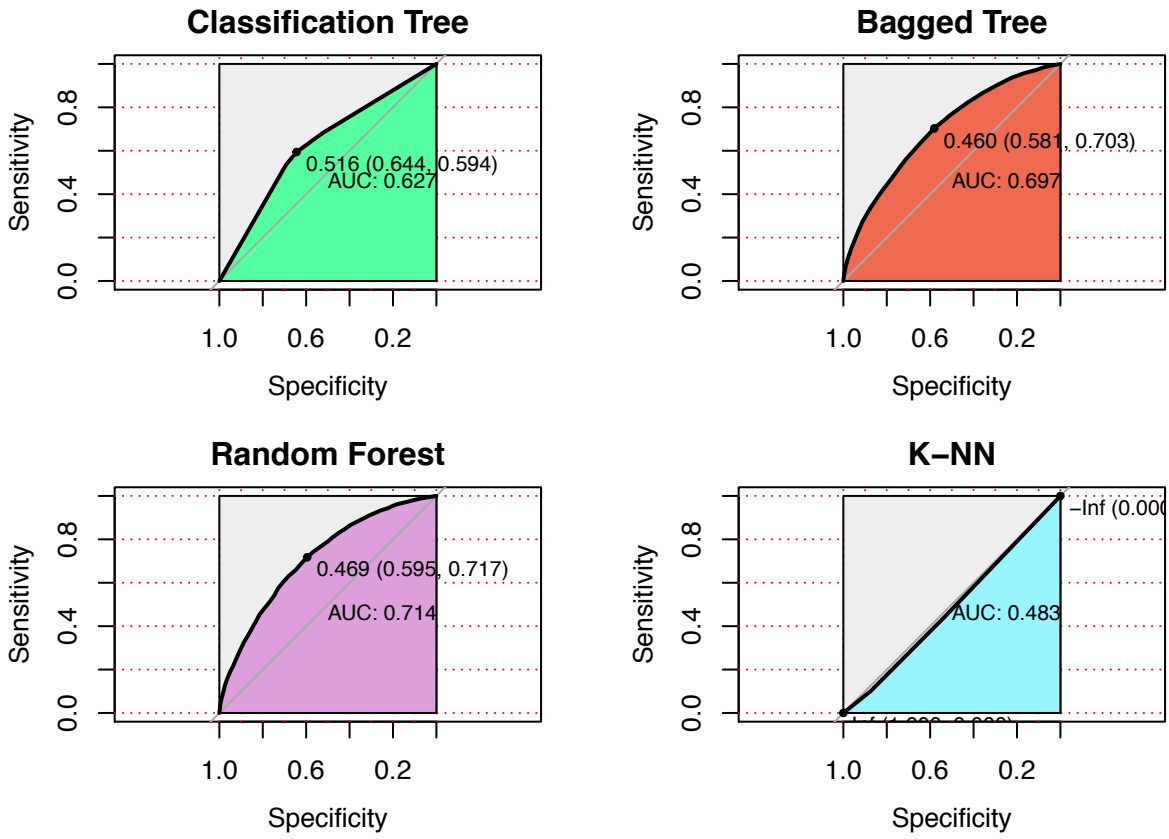
```
baggedtree.roc <- roc(testSet$shares, bag.model.prob[,2])
plot(baggedtree.roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("red", "red"), max.auc.polygon=TRUE,
     auc.polygon.col='coral2', print.thres=TRUE, main="Bagged Tree")
```

```
##
## Call:
## roc.default(response = testSet$shares, predictor = bag.model.prob[,     2])
##
## Data: bag.model.prob[, 2] in 5990 controls (testSet$shares 0) < 5903 cases (testSet$shares 1).
## Area under the curve: 0.6965
```

```
rf.roc <- roc(testSet$shares, rf.model.prob[,2])
plot(rf.roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("red", "red"), max.auc.polygon=TRUE,
     auc.polygon.col='plum', print.thres=TRUE, main="Random Forest")
```

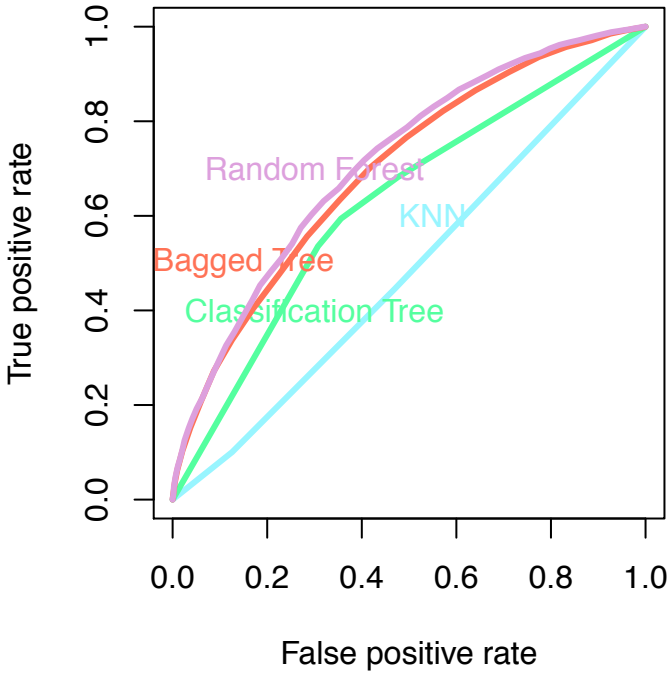
```
##
## Call:
## roc.default(response = testSet$shares, predictor = rf.model.prob[,     2])
##
## Data: rf.model.prob[, 2] in 5990 controls (testSet$shares 0) < 5903 cases (testSet$shares 1).
## Area under the curve: 0.7136
```

```
knn.roc <- roc(testSet$shares, attributes(webData.knn.prob)$prob )
plot(knn.roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1, 0.2),
     grid.col=c("red", "red"), max.auc.polygon=TRUE,
     auc.polygon.col='cadetblue1', print.thres=TRUE, main="K-NN")
```




```
##
## Call:
## roc.default(response = testSet$shares, predictor = attributes(webData.knn.prob)$prob)
##
## Data: attributes(webData.knn.prob)$prob in 5990 controls (testSet$shares 0) < 5903 cases (testSet$sha
## Area under the curve: 0.4835
```

```
# COMPARISON PLOT

ROCCurve<-par(mfrow=c(1,1), pty = "s")
plot(performance(prediction(attributes(webData.knn.prob)$prob, testSet$shares),'tpr','fpr'),
     col="cadetblue1", lwd=3
)
text(0.55,0.6,"KNN",col="cadetblue1")
plot(performance(prediction(classmodel.tree.prob[,2], testSet$shares),'tpr','fpr'),
     col="seagreen1", lwd=3, add=TRUE
)
text(0.3,0.4,"Classification Tree",col="seagreen1")
plot(performance(prediction(bag.model.prob[,2], testSet$shares),'tpr','fpr'),
     col="coral1", lwd=3, add=TRUE
)
text(0.15,0.5,"Bagged Tree",col="coral1")
plot(performance(prediction(rf.model.prob[,2], testSet$shares),'tpr','fpr'),
     col="plum", lwd=3, add=TRUE
)
text(0.3,0.7,"Random Forest",col="plum")
```


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Random Forest
KNN
Bagged Tree
Classification Tree
False positive rate

Attribute Information:

Number of Attributes: 61 (58 predictive attributes, 2 non-predictive, 1 goal field)

Attribute Information:
0. url: URL of the article (non-predictive)
1. timedelta: Days between the article publication and the dataset acquisition (non-predictive)
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)