Online News Popularity Data Set - Final Project Report

CIS/CSE400/600/691

Fundamentals of Data Mining and Information Management

Ishan Gupta: MSCE, Syracuse University, igupt100@syr.edu

Rohit Sharma: MSCE, Syracuse University, rshar102@syr.edu

Saurabh Agrawal: MSCE, Syracuse University, skagrawa@syr.edu

Varun Jindal: MSCE, Syracuse University,vjindal@syr.edu

Abstract

Over the past few decades Internet has drastically changed the way ideas, stories, facts are propagated across readership. We as a society have evolved technologically. From Books to Cinema, Newspaper to News Broadcasts and Magazines to Blogs. It's far easier now to voice your opinions/ideas then it could ever have been before. Having said that, although it's easier to publish articles online with little or no cost involved, yet finding the readers for it is an altogether different task. How and why does an article get popular is the real mystery.

In this study, we intend to use a dataset to identify the criteria that make articles popular. We also intend to do a comparative study of various Data Mining algorithms on 'Online News Popularity Dataset' and see how they fair with each other in determining whether an article will be popular or not.

Introduction

Online News Popularity Data Set acquired from University of California Irvine's Machine Learning Repository summarizes a heterogeneous set of features/attributes about articles published by Mashable in a period of two years.

The Goal of this Project is to analyze this data set and predict the Popularity of an article based on the number of shares made for the (url containing article) across various social networks.

The dataset comprises of 61 attributes of which 2 are non-predictive, 58 numeric attributes of predictive nature, and one is a class label.

A first look at data reveals minor aberrations. There are few outliers and none 'NA' attribute values. The idea is to preprocess the data to remove these very outliers. Once that Data is clean, we divide the dataset into training and testing data. Then we train the models on training data using various classification methods. In this Project, we have taken up 4 different classification methods to train models using dataset viz. CART, C5.0, KNN, and Random Forest.

These trained models are then used to perform classification on testing data. Results are recorded by plotting a confusion Matrix. The Model with highest accuracy provides the most efficient result. It's important to project the outcomes of various models through a Visualization form. We, here, have used ROC curve to depict the efficiencies of different training methods used.

Data Set Description

Following features are of critical importance when understanding the data set:

1. It is a Multivariate data set as is evident from the Number of Attributes which is 61.
2. The Attributes are all Integer or Real barring URL and time delta which are Non-Predictive in nature.
3. Total Number of Rows/Instance: 39797
4. Total Number of Columns/Attributes: 61

NOTE: Although the data set comprises of ~40000 attributes. We have sampled out 15000 Instances to be used for our analysis. This was an important modification to be able to train data under Random Forest algorithm. Random Forest requires a huge amount of memory to generate trees that can be efficient enough to predict. Using the data set as it is would repeatedly result in machine "Out of Memory" error. We tried to work around this issue by transporting data to more efficient machines and to tweak with number of trees and nodesize parameters in Random Forest but it didn't produce expected results. The most efficient way to overcome this shortcoming we found was to reduce the data set to a lesser, more viable number.

Data Set Characteristics:	Multivariate	**Number of Instances:**	39797	**Area:**	Business
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	61	**Date Donated**	2015-05-31
Associated Tasks:	Classification, Regression	**Missing Values?**	N/A	**Number of Web Hits:**	55476

Attributes:

The attributes contained in the data set are described below:

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)

Data Analysis

Goal: The main objective here is to classify the dataset into popular or not popular category by training models on training data.

The first and foremost step for any Data Mining Task is to understand the data. Here is a brief snapshot of a few predictive attributes and the Class/Target Attribute i.e. “shares”.

We arrive at this view by getting the head of the dataset.

- It's evident by taking a look at the dataset that 'URL' and 'timedelta' aren't the predictive attributes and will not reveal any interesting information in further predictions. Keeping this in mind we shall remove those attributes while cleaning the Data.
- Further it's interesting to note the varying values for shares. It's important to come up with a measure to be able to base the Popularity of an article on a pragmatic value.

Data Preprocessing

- The Dataset comprises of 39645 observations. In order to be able to train data using various models we get a sample of 15000 (10000 training records, 5000 testing records) randomized observation. Summarizing the data reveals a few outliers for several attributes. It also reveals the median for "shares" which is equal to 1500.
- Next, we remove "url" and "timedelta" (two non-predictive) attributes from the Dataset.
- Generate zscore using the scale function to standardize the dataset.
- We further remove all the outliers from the dataset. Interestingly there is only one observation that can be considered an outlier, which is n_unique_tokens =701.
- Set shares to 1 if greater than the median value i.e. >1400 else 0. Here 1 stands for Popular and 0 stands for Not Popular.
- We now have a clean dataset which can be used to train models using different algorithms.
- We divide the data set into training and testing data sets.

	url	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words
1	http://mashable.com/2013/01/07/amazon-instant-v...	731	12	219	0.6635945	1
2	http://mashable.com/2013/01/07/ap-samsung-spo...	731	9	255	0.6047431	1
3	http://mashable.com/2013/01/07/apple-40-billion-a...	731	9	211	0.5751295	1
4	http://mashable.com/2013/01/07/astronaut-notre-d...	731	9	531	0.5037879	1
5	http://mashable.com/2013/01/07/att-u-verse-apps/	731	13	1072	0.4156456	1
6	http://mashable.com/2013/01/07/beewi-smart-toys/	731	10	370	0.5598886	1

abs_title_subjectivity	abs_title_sentiment_polarity	shares
0.00000000	0.1875000	593
0.50000000	0.0000000	711
0.50000000	0.0000000	1500
0.50000000	0.0000000	1200
0.04545455	0.1363636	505
0.14285714	0.2142857	855

```
#read data
news<-read.csv
("OnlineNewsPopularity.csv")
#summary news dataset
summary (news)
#Data Preprocessing
#remove url and timedelta from the dataset using subset
newsSubset <- subset( news, select = -c(url, timedelta ) )
```

```
# generate z-scores using the scale() function
for(i in ncol(newsSubset)-1){
  newsSubset[,i]<-scale(newsSubset[,i],
center = TRUE, scale = TRUE)
}
#sample data before classifying dataset into
training and test
set.seed(100)
# Dataset for classification
newsdataset <-newsSubset[1:15000,]
#set shares to 1 if greater than 1400 else 0
newsdataset$shares                    <-
as.factor(ifelse(newsdataset$shares
>1400,1,0))
#training dataset observation 1 to 10000
newstrain <- newsdataset[1:10000,]
#test dataset observation 10001 to 15000
newstest <- newsdataset[10001:15000,]
```

Data Mining Methods

Now that we have a clean Dataset. We start by training different models and identifying the ones that reveal the best accuracy. Before Training the model, it's important to identify the fact that we are here dealing with a large number of attributes. Thus for effective accuracy we need to train models using algorithms that support large number of attributes. Over considerable tests we finally shortlisted 4 different algorithms. CART, C50, KNN and Random Forest.

We shall now look at how all these algorithms fair one by one.

1. KNN or K-nearest Neighbor

The KNN or *k*-nearest neighbor algorithm is a type of instance-based learning, where new data are classified based on stored, labeled instances. Considering the fact that our dataset attributes have values that do not skew a lot, it becomes our first choice for developing a trained model.

i) We work around with different values for k to find the one that best fits our model.

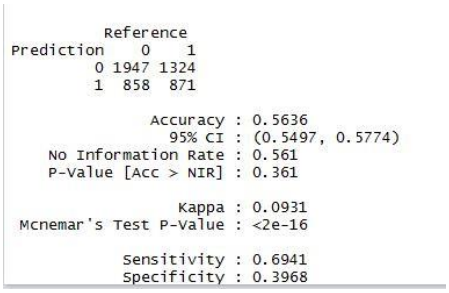
ii) It's important to keep in mind since we do not have skewed values a large number for 'k' doesn't make a drastic difference. But still to test this theory we work with K values of 1, 3, 5 and 10

iii) K=3 reveals the highest accuracy amongst its counterparts.

iv) We developed confusion Matrix for All values to validate the accuracy for every trained Model for different values of K.

```
#train using knn3
newsknn3 <- knn3(shares ~.,newstrain)
#predict
newsknn3class<-predict(newsknn3,
newstest, type="class")
newsknn3prob<-predict(newsknn3,
newstest,type="prob")
# Confusion matrix
confusionMatrix(newsknn3class,
newstest$shares)
```

Confusion Matrix for KNN3



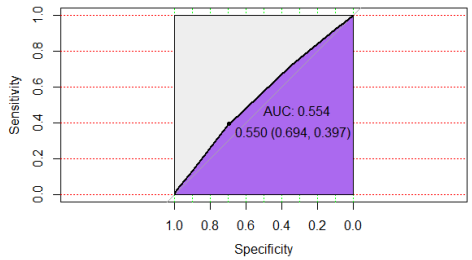
```
          Reference
Prediction    0    1
         0 1947 1324
         1  858  871

               Accuracy : 0.5636
                 95% CI : (0.5497, 0.5774)
    No Information Rate : 0.561
    P-Value [Acc > NIR] : 0.361

                  Kappa : 0.0931
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6941
            Specificity : 0.3968
```

ROC Curve for KNN3




```
          Reference
Prediction    0    1
         0  903  354
         1 1902 1841

               Accuracy : 0.5488
                 95% CI : (0.5349, 0.5627)
    No Information Rate : 0.561
    P-Value [Acc > NIR] : 0.9601

                  Kappa : 0.1492
 Mcnemar's Test P-Value : <2e-16
```

2. **CART- Classification and Regression Trees**

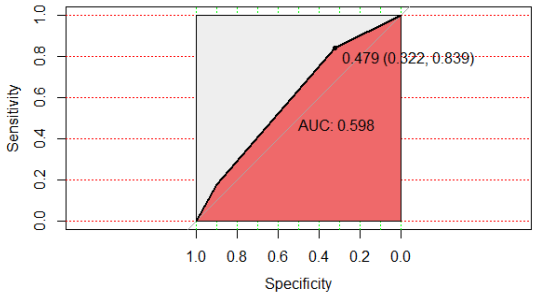
Classification and Regression Trees or CART can be used for logistic regression. The dataset we are working here with requires a better training model than the lazy learning KNN model. Our next approach here is to train the models using CART package under rpart library.

i) CART forms an intelligent decision tree that improves the overall accuracy of the training model.

ii) Since CART can be used to classify and predict the class label and also to identify any numerical values through Regression, it fits the kind of model we intend for predicting our class label. "Shares" is both the Target Attribute and has Numerical value of either 1 or 0.

```
newsCart<-rpart(shares ~.,newstrain,method
='class')
fancyRpartPlot(newsCart)
newsCartClassPrediction<-predict(
newsCart,newstest ,type="class")
newsCartProbPrediction<-predict(
newsCart,newstest ,type="prob")
# Confusion matrix for CART
confusionMatrix(newsCartClassPrediction,
newstest$shares)
```

ROC Curve for CART



3. **C5.0- Decision Tree**

So far we have worked on developing training Models using KNN and CART methods. The accuracy in either cases has been ordinary. Next we develop a training model using C5.0-Decision Trees. C5.0 is decision trees and rule-based models for Predictions.

i) C5.0 models are rule based and they work on intelligently retaining the predictions and results as they go over the training data recursively.

ii) The number of trials have an adverse effect on the speed of the training model but with greater trials the accuracy has a marked change.

iii) We iterated the training model for different trial values starting from 5 and going all the way to 12.

iv)The model showed significant improvement until trial=9 beyond which it plateaued.

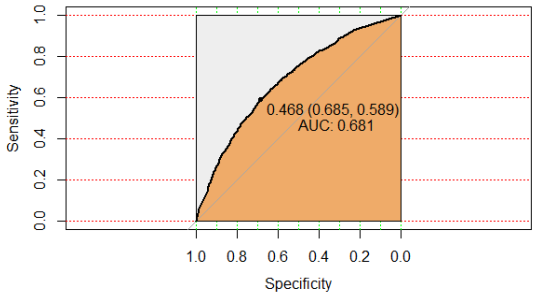
Confusion Matrix for C5.0 for Trial=9

```
          Reference
Prediction    0    1
         0 1960  942
         1  845 1253

               Accuracy : 0.6426
                 95% CI : (0.6291, 0.6559)
    No Information Rate : 0.561
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.2709
 Mcnemar's Test P-Value : 0.02315
```

ROC Curve for C5.0



4. Random Forest

Random Forest classifier uses a number of decision trees, in order to improve the classification rate. So far, C5.0 decision based trees have shown promising accuracy figures. Random Forest is a type of an ensemble method since it builds multiple decision trees whose outputs are averaged together for obtaining a final output to improve the efficacy of the model at predictions.

i) We worked on developing different models trained using Random Forest.

ii) The approach was to have a smaller node size so as to improve the accuracy. A lower node size results in a huge decision tree which has the drawback of being slow but is better at predicting the correct class label for the Target Attributes

iii) We trained three different models with node size of 10, 15 and 20 and combined them to produce a combined output using an ensemble of random forest models.

iv) The Accuracy achieved by this model was highest amongst all other models.

```
#combining all randomForest trained models
rf.all<-combine(newsrf10,
newsrf20,newsrf15)
newspredallclass<-predict( rf.all,newstest,
type="class")
confusionMatrix(newspredallclass,
newstest$shares)
```

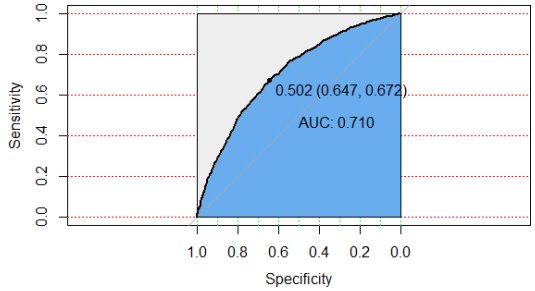
Confusion Matrix Random Forest

```
          Reference
Prediction    0    1
         0 1801  712
         1 1004 1483

               Accuracy : 0.6568
                 95% CI : (0.6435, 0.67)
    No Information Rate : 0.561
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3132
 Mcnemar's Test P-Value : 2.144e-12
```

ROC Curve for Random Forest



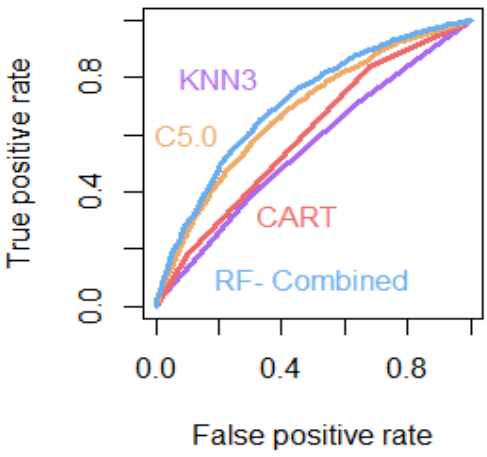
Conclusion

A comparative study by training different classification models on the dataset and using them to predict output class on testing data has revealed that Random Forest is the best model among all considered models for an accurate prediction of the popularity of an online news article.

Random Forest generates multiple trees and predicts final class output after taking output of each one of them into account and hence provides far more accurate and conclusive predictions as compared to other models.

The shortcoming thus far has been the inability to affectively train RF models with lesser node sizes. This is more so a technical problem and can be overcome by meeting the memory requirements for the random forest algorithm or by fine tuning the trained model using some carefully determined parameter values. We tweaked with various input parameters while training the model and settled with reduced number of records to get the best accuracy. However that's not the only way and our future goal is to increase the performance of Random Forest by finding the optimum values of other parameters without having to reduce the dataset. Final accuracy results are as below:
KNN - 56.36%
CART - 54.88%
C5.0 - 64.26%
RF - 65.68%

Following ROC plot depicts comparative performance of all four classification models.



References

Source Dataset:
https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity

Additional References:

- *https://en.wikipedia.org/wiki/Decision_tree_learning*
- *https://en.wikipedia.org/wiki/CART*
- *https://en.wikipedia.org/wiki/Random_forest*
- *https://cran.r-project.org/web/packages/C50/C50.pdf*
- *http://yilinwei.com/project/Online-news-popularity-classification-with-R.html*
- *https://cran.r-project.org/web/packages/randomForest/randomForest.pdf*
- *https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf*
- *https://cran.r-project.org/web/packages/class/class.pdf*