Capstone Project

Jenifer Jones

Thursday, June 04, 2015

```
## Loading required package: NLP
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
## Loading required package: lattice
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
##
## The following object is masked from 'package:NLP':
##
##     annotate
##
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:randomForest':
##
##     combine
##
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Introduction

The purpose of this document is to satisfy the Capstone Project for completion of the Data Science Workshop through SlideRule. The Data Science Workshop is a collection of courses related to the field with the intention of providing a good foundation in Data Science.
For the Capstone Project I have elected to participate in a Kaggle Competition related to the Analytics Edge course offered through edX by MIT. This competition used data about New York Times blots articles and the purpose was to create a model which would predict

if an article would be popular or not based on the variables provided. Once submitted the models were evaluated using AUC, which is a commonly used metric for binary classification problems. It provides the proportion of time that the model predicted the correct value.
This competition ran for three weeks from April 14, 2015 - May 4, 2015.

Data Set Outline

Two datasets were provided, a training set with 6,532 articles and a test set with 1,870 articles. The dependent variable is Popular which is binary and set to 1 if an article had 25 or more comments in its online comment section. There were 8 independent variables in the datasets and are as follows.

- NewsDesk = the New York Times desk that produced the story
- SectionName = section the article appeared in
- SubsectionName = subsection the article appeared in
- Headline = title of the article
- Snippet = small portion of article text
- Abstract = summary of blog article, written by the New York Times
- WordCount = number of words in the article
- PubDate = Publication Date
- UniqueID = unique identifier for each article

(source: https://www.kaggle.com/c/15-071x-the-analytics-edge-competition-spring-2015/data)

Exploratory Data Analysis

The purpose of Exploratory Data Analysis (EDA) is to allow the analyst to understand the data set and start to gauge what may or may not be related.
During this phase the analyst will often create plots, calculate statistics and understand which variables could be manipulated for use in the model.

To start it is important to get a sense of what the underlying data looks like, first looking at the percentage of the records within the training set that are noted as being popular. This gives us a baseline accuracy that we can work from, if we assume that all predictions are popular.

```
#Load baseline file
NYT_train = read.csv("NYTimesBlogTrain.csv", stringsAsFactors=FALSE)
NYT_test = read.csv("NYTimesBlogTest.csv", stringsAsFactors=FALSE)

sum(NYT_train$Popular == 1)/nrow(NYT_train)

## [1] 0.1673301
```

From this we see that a very small number of the articles, approximately 17% are considered popular.

What does a summary of all the columns look like? Are we missing values? Are there any that look to be the wrong data type?

```
summary(NYT_train)
```

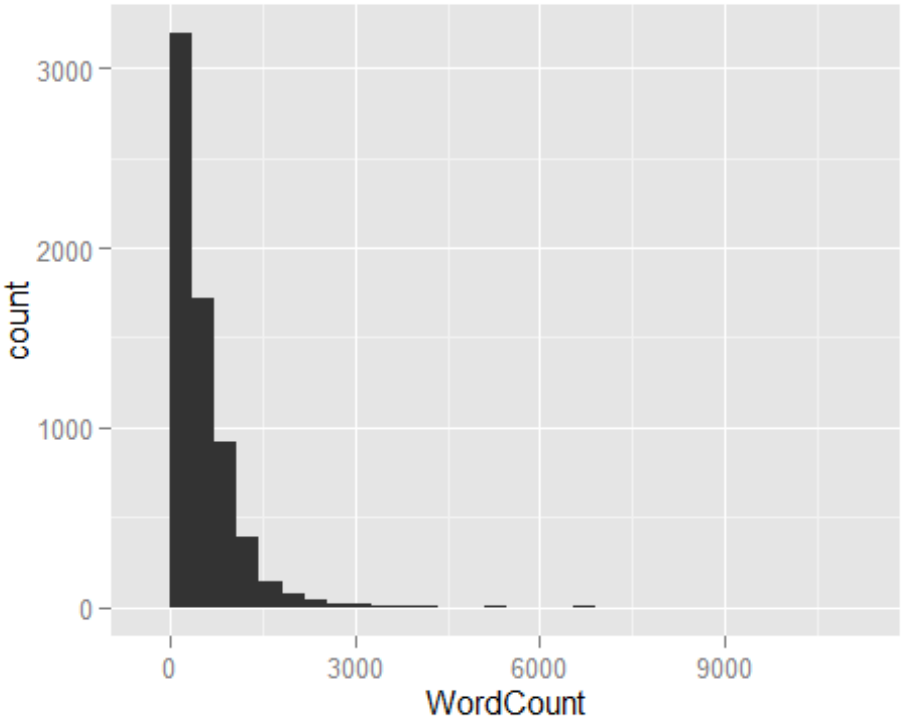
```
##    NewsDesk         SectionName        SubsectionName
##  Length:6532        Length:6532        Length:6532
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##    Headline           Snippet            Abstract
##  Length:6532        Length:6532        Length:6532
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##    WordCount         PubDate              Popular          UniqueID
##  Min.   :    0.0   Length:6532        Min.   :0.0000   Min.   :   1
##  1st Qu.:  187.0   Class :character   1st Qu.:0.0000   1st Qu.:1634
##  Median :  374.0   Mode  :character   Median :0.0000   Median :3266
##  Mean   :  524.4                      Mean   :0.1673   Mean   :3266
##  3rd Qu.:  723.2                      3rd Qu.:0.0000   3rd Qu.:4899
##  Max.   :10912.0                      Max.   :1.0000   Max.   :6532
```

There are a couple of points from the summary that guide us to our next steps. One is that publication date is a character variable. This needs to be changed to a date. Second is that all of the section variables are character, these should be changed to factors. Finally there is the large spread in the word count variable, the maximum value is quite a bit larger than where the third quartile is noted. It may be worth creating a histogram on word count to see what the distribution of the value looks like.

```
#Converting Publication Date to a date/time variable
NYT_train$PubDate = strptime(NYT_train$PubDate, "%Y-%m-%d %H:%M:%S")
NYT_test$PubDate = strptime(NYT_test$PubDate, "%Y-%m-%d %H:%M:%S")
#Add a new column with the factor variable to indicate the day of the week
NYT_train$DayofWeek=weekdays(as.Date(NYT_train$PubDate, "%Y-%m%d"))
NYT_test$DayofWeek=weekdays(as.Date(NYT_test$PubDate, "%Y-%m%d"))
#Add a new column based on the result of a logical check to indicate weekend
or weekday
NYT_train$DayType=ifelse(NYT_train$DayofWeek=="Sunday" | NYT_train$DayofWeek
== "Saturday",
                         "Weekend", "Weekday")
NYT_test$DayType=ifelse(NYT_test$DayofWeek=="Sunday" | NYT_test$DayofWeek ==
"Saturday",
```

```
                          "Weekend", "Weekday")
#Add a column to indicate the hour of the publication
NYT_train$Hour=hour(NYT_train$PubDate)
NYT_test$Hour=hour(NYT_test$PubDate)
#Converting NewsDesk/Section variables to factors
#NYT_train$NewsDesk = as.factor(NYT_train$NewsDesk)
##NYT_train$SectionName = as.factor(NYT_train$SectionName)
#NYT_train$SubsectionName = as.factor(NYT_train$SubsectionName)
#NYT_test$NewsDesk = as.factor(NYT_test$NewsDesk)
#NYT_test$SectionName = as.factor(NYT_test$SectionName)
#NYT_test$SubsectionName = as.factor(NYT_test$SubsectionName)

## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust
this.
```



Looking at the histogram, we can see that the large majority of the articles are less than 1000 words in length. Is there a difference in the popularity depending on the word count.

```
table(subset(NYT_train, NYT_train$WordCount<=187)$Popular)
```

From this we see that the smaller articles are less likely to be popular, 4.7% vs the baseline of 16.73%.

```
table(subset(NYT_train, NYT_train$WordCount>723)$Popular)
```

The reverse is true for the longer articles, those that are over the third quartile value are over twice as likely to be popular than our baseline value.

As mentioned above the spread between the minimum and maximum is quite large, however based on the graph the number of articles over 2000 words is very small. This indicates that word count will be a key factor in determining the popularity of an article.

```
#Assign a factor column to indicate small (<=187 words), medium (between 187
and 723), large (>723)
NYT_train$ArticleSize = as.factor(ifelse(NYT_train$WordCount>723, "Large",
                          ifelse(NYT_train$WordCount<=187, "Small",
"Medium")))
NYT_test$ArticleSize = as.factor(ifelse(NYT_test$WordCount>723, "Large",
                                        ifelse(NYT_test$WordCount<=187,
"Small", "Medium")))
table(NYT_train$ArticleSize, NYT_train$Popular)
```

Another items to look for is missing values as these may be able to be derived from other fields or imputed using standard techniques.

```
summary(NYT_train$NewsDesk)

##    Length     Class      Mode
##      6532 character character
```

It looks like there are 1846 articles that aren't assigned a News Desk, looking at a table of the News Desk and Section Name we can see that there is a 1 to 1 relationship between them so we can infer the missing values for both variables based on this.

```
#Set blank "Businesss Day" to NewsDesk = "Business"
NYT_train$UpdatedNewsDesk<-NYT_train$NewsDesk

NYT_train$UpdatedNewsDesk = ifelse((NYT_train$SectionName == "Business Day"),
"Business",
                                   NYT_train$UpdatedNewsDesk)
#Set blank "Crosswords/Games" to NewsDesk = "Business"
NYT_train$UpdatedNewsDesk=ifelse((NYT_train$SectionName ==
"Crosswords/Games"), "Business",
                                 NYT_train$UpdatedNewsDesk)
#Set blank "Health" to NewsDesk = "Science"
NYT_train$UpdatedNewsDesk=ifelse((NYT_train$SectionName == "Health"),
"Science",
                                 NYT_train$UpdatedNewsDesk)
#Set blank "Multimedia" to NewsDesk = "Multimedia"(new value since all are
blank)
NYT_train$UpdatedNewsDesk=ifelse((NYT_train$SectionName == "Multimedia"),
"Multimedia",
                                 NYT_train$UpdatedNewsDesk)
#Set blank "Opinion" to NewsDesk = "OpEd"
NYT_train$UpdatedNewsDesk=ifelse((NYT_train$SectionName == "Opinion"),
"OpEd",
                                 NYT_train$UpdatedNewsDesk)
#Set blank "Travel" to NewsDesk = "Travel"
NYT_train$UpdatedNewsDesk=ifelse((NYT_train$SectionName == "Travel"),
```

```
"Travel",
                                    NYT_train$UpdatedNewsDesk)
#Set blank "U.S." to NewsDesk = "Styles"
NYT_train$UpdatedNewsDesk=ifelse((NYT_train$SectionName == "U.S."), "Styles",
                                    NYT_train$UpdatedNewsDesk)

#Now the reverse
#Set for Section Name
NYT_train$UpdatedSectionName<-NYT_train$SectionName
#Tstyle = Tstyle
NYT_train$UpdatedSectionName = ifelse((NYT_train$UpdatedNewsDesk ==
"TStyle"), "TStyle",
                                        NYT_train$UpdatedSectionName)
#Foreign = World
NYT_train$UpdatedSectionName = ifelse((NYT_train$UpdatedNewsDesk ==
"Foreign"), "World",
                                        NYT_train$UpdatedSectionName)
#Styles = U.S.
NYT_train$UpdatedSectionName = ifelse((NYT_train$UpdatedNewsDesk ==
"Styles"), "U.S.",
                                        NYT_train$UpdatedSectionName)
#Business = Business Day
#Culture = Arts
NYT_train$UpdatedSectionName = ifelse((NYT_train$UpdatedNewsDesk ==
"Culture"), "Arts",
                                        NYT_train$UpdatedSectionName)
#National = U.S.
NYT_train$UpdatedSectionName = ifelse((NYT_train$UpdatedNewsDesk ==
"National"), "U.S.",
                                        NYT_train$UpdatedSectionName)
#oped = opinion
NYT_train$UpdatedSectionName = ifelse((NYT_train$UpdatedNewsDesk == "OpEd"),
"Opinion",
                                        NYT_train$UpdatedSectionName)
#Science = Health
NYT_train$UpdatedSectionName = ifelse((NYT_train$UpdatedNewsDesk ==
"Science"), "Health",
                                        NYT_train$UpdatedSectionName)
#Sports = Sports
NYT_train$UpdatedSectionName = ifelse((NYT_train$UpdatedNewsDesk ==
"Sports"), "Sports",
                                        NYT_train$UpdatedSectionName)
NYT_train$UpdatedNewsDesk[NYT_train$UpdatedNewsDesk==""]<-"No News Desk"
NYT_train$UpdatedSectionName[NYT_train$UpdatedSectionName==""]<-"No Section"
```

The same set of steps were completed for the test set.

Before moving on to model creation there may be some meaning in the types of words a headline starts with which cause people to comment. For example, if it asks a question or asks readers to respond then that might lead to more comments. At this point some offline

text review was done in the CSV file directly to understand if this assumption was valid, from the review done it looks like it was valid and new factors can be assigned.

```
#Develop vectors of "question words" and assign to "QuestionWord" column
which will be 1, 0
TwoLetter=c("If ", "Is ", "Do ")
ThreeLtr=c("Ask ", "Are ", "Can ", "Why ", "Has ", "How ", "Who ")
FourLtr=c("Does ", "What ", "Will ", "Have ", "When ")
FiveLtr=c("Could ", "Would ", "Where ")
SixLtr=c("Should ")
NYT_train$QuestionWord=0
NYT_train$QuestionWord=ifelse(NYT_train$QuestionWord==1, 1,
ifelse(substr(NYT_train$Headline,1,3) %in% TwoLetter, 1, 0))
NYT_train$QuestionWord=ifelse(NYT_train$QuestionWord==1, 1,
ifelse(substr(NYT_train$Headline,1,4) %in% ThreeLtr, 1, 0))
NYT_train$QuestionWord=ifelse(NYT_train$QuestionWord==1, 1,
ifelse(substr(NYT_train$Headline,1,5) %in% FourLtr, 1, 0))
NYT_train$QuestionWord=ifelse(NYT_train$QuestionWord==1, 1,
ifelse(substr(NYT_train$Headline,1,6) %in% FiveLtr, 1, 0))
NYT_train$QuestionWord=ifelse(NYT_train$QuestionWord==1, 1,
ifelse(substr(NYT_train$Headline,1,7) %in% SixLtr, 1, 0))

NYT_train$CommentRequested=ifelse(substr(NYT_train$Headline, 1, 20) == "No
Comment Necessary",1,0)
NYT_train$CommentRequested=ifelse(NYT_train$CommentRequested==1, 1,
                                  ifelse(substr(NYT_train$Headline,1,15) ==
"Readers Respond" , 1, 0))

NYT_train$NoComment =ifelse(substr(NYT_train$Headline, 1, 9) == "Q. and
A.",1,0)
NYT_train$NoComment=ifelse(NYT_train$NoComment==1, 1,
                                  ifelse(substr(NYT_train$Headline,1,13) ==
"Test Yourself" , 1, 0))
```

The same column was added to the test set.

The last feature to be added before starting model creation is the headline length.

```
NYT_train$HeadlineLength = nchar(NYT_train$Headline)
NYT_test$HeadlineLength = nchar(NYT_test$Headline)
```

Model Creation

In order to determine the best model for submission multiple types of models were analyzed with different variables. The two that will be outlined below are logistic regression and Random Forest.

Logistic Models

The first model created used the following variables: updated News Desk and Section Name, Type of Day (i.e. WeekDay vs Week End), Hour of Day and Article Size. This model scored a Public AUC value of 0.90002 when submitted to the Kaggle site.

```
Model1=  glm(Popular ~ UpdatedNewsDesk  + UpdatedSectionName + DayType + Hour
+ ArticleSize,
             data=NYT_train, family=binomial)
PredTest = predict(Model1, newdata=NYT_test, type="response")
```

Based on the summary we can see that the variables chosen are significant, with some factor values being more significant than others. Based on the outcomes from this model it will be used as the baseline to measure how adding variables changes the outcome.

Building on the success of the first model, we will use the same values with the addition of the question word indicator as well as the factors for comment requested and no comment.

```
Model2=  glm(Popular ~ UpdatedNewsDesk  + UpdatedSectionName + DayType + Hour
+ ArticleSize +
               QuestionWord + CommentRequested + NoComment,  data=NYT_train,
family=binomial)
PredTest = predict(Model2, newdata=NYT_test, type="response")
```

Based on the results from the public site our AUC score increased to 0.90152. While this value does not seem very large, the incremental increase put the model in the top 25% of the class at the time.

To finish off the logistic model testing one more model was tested, this included the addition of headline length. Unfortunately, this did not make the model more successful, in fact it decreased the public AUC score to 0.56

```
Model3=  glm(Popular ~ UpdatedNewsDesk  + UpdatedSectionName + HeadlineLength
+ DayType +
               Hour + ArticleSize + QuestionWord + CommentRequested +
NoComment,
             data=NYT_train, family=binomial)
PredTest = predict(Model3, newdata=NYT_test, type="response")
```

Random Forest Models

Two simple random foreest models were created using the same variables as those in the logistic models above however neither was an increase in public AUC score, in fact both were closer to the value from the Headline Length submission of 0.56.

In order to create a random forest model that will be different than the logistic regression model we will use some text analytics techniques.

First we will use the "Snippet" variable to create a text corpus. The training and testing datasets need to be combined prior to creating the corpus so that all terms are recognized

within the model. We will run through the standard techniques of changing all the case to lower case, removing punctuation and removing standard English stop words. Once those steps are done a Document Term Matrix is created, for our model we will remove sparse terms so that we are only using words that are commonly found in the snippets. After this is completed the data sets need to be split back into training and test sets and the other variable columns added back in. Note that for this model we will be using the log of Word Count rather than the factorized Article Size that was used in the logistic regression models.

```
CorpusSnippet = Corpus(VectorSource(c(NYT_train$Snippet, NYT_test$Snippet)))
CorpusSnippet = tm_map(CorpusSnippet, tolower)
# Remember this extra line is needed after running the tolower step:
CorpusSnippet = tm_map(CorpusSnippet, PlainTextDocument)
CorpusSnippet = tm_map(CorpusSnippet, removePunctuation)
CorpusSnippet = tm_map(CorpusSnippet, removeWords, stopwords("english"))
CorpusSnippet = tm_map(CorpusSnippet, stemDocument)

dtm = DocumentTermMatrix(CorpusSnippet)
sparse = removeSparseTerms(dtm, 0.995)
SnippetWords = as.data.frame(as.matrix(sparse))
# Let's make sure our variable names are okay for R:
colnames(SnippetWords) = make.names(colnames(SnippetWords))
#Split the observations back into the training set and testing set.
SnippetWordsTrain = head(SnippetWords, nrow(NYT_train))
SnippetWordsTest = tail(SnippetWords, nrow(NYT_test))

SnippetWordsTrain$Popular = NYT_train$Popular

SnippetWordsTrain$LogWord = log(1 + NYT_train$WordCount)
SnippetWordsTest$LogWord = log(1 + NYT_test$WordCount)

SnippetWordsTrain$UpdatedNewsDesk = as.factor(NYT_train$UpdatedNewsDesk)
SnippetWordsTest$UpdatedNewsDesk = as.factor(NYT_test$UpdatedNewsDesk)

SnippetWordsTrain$UpdatedSectionName = NYT_train$UpdatedSectionName
SnippetWordsTest$UpdatedSectionName = NYT_test$UpdatedSectionName

SnippetWordsTrain$DayType = NYT_train$DayType
SnippetWordsTest$DayType = NYT_test$DayType

SnippetWordsTrain$Hour = NYT_train$Hour
SnippetWordsTest$Hour = NYT_test$Hour

SnippetWordsTrain$QuestionWord = NYT_train$QuestionWord
SnippetWordsTest$QuestionWord = NYT_test$QuestionWord

SnippetWordsTrain$CommentRequested = NYT_train$CommentRequested
SnippetWordsTest$CommentRequested = NYT_test$CommentRequested
```

```
SnippetWordsTrain$NoComment = NYT_train$NoComment
SnippetWordsTest$NoComment = NYT_test$NoComment

SnippetWordsTrain$HeadlineLength = NYT_train$HeadlineLength
SnippetWordsTest$HeadlineLength = NYT_test$HeadlineLength

SnippetWordsTrain$RelatedNewsDesk = NYT_train$RelatedNewsDesk
SnippetWordsTest$RelatedNewsDesk = NYT_test$RelatedNewsDesk

SnippetWordsTrain$DayTypeFctr = NYT_train$DayTypeFctr
SnippetWordsTest$DayTypeFctr = NYT_test$DayTypeFctr

SnippetWordsTrain$DayTypeFctr = NYT_train$DayTypeFctr
SnippetWordsTest$DayTypeFctr = NYT_test$DayTypeFctr
```

Once all of the text data is ready we will use k-fold cross validation with the random forest model. Based on the results from this model the public AUC score was 0.83946. While this is lower than the other models from logistic regression it uses different variables and still has significance.

```
tr.control = trainControl(method="cv", number = 10)
cp.grid = expand.grid( .cp = (0:10)*0.001)
tr = train(as.factor(Popular) ~ ., data = SnippetWordsTrain, method =
"rpart",
           trControl = tr.control, tuneGrid = cp.grid)
best.tree=tr$finalModel
PredTest = predict(best.tree, newdata=SnippetWordsTest, type = "prob")
```

Deciding on Best Submission

In order to create a submission that was representative of different techniques and variables I combined the best Logistic and Random Forest models together to create one submission with the predictions from each weighted at 50%. When submitted the public AUC score was 0.88920. This value is less than the logistic regression model alone, however since the public score is not based on the full test data set it was selected as it had more variables taken into consideration.

Conclusions/Thoughts on Further Analysis

At the end of the competition my combined model scored 0.87385 which put me in the top 65% of the class. Some of the thoughts I had on further analysis include:

- Reviewing more of the text within the articles to find key words, however this can be misleading if the data set is time specific as some articles that are popular today may not show up in the future (for example, ebola articles)
- Using more advanced modeling techniques such as Bayesian algorithms

- Additional factorization, the hour of day was used here, but maybe grouping it into time of day would have been more relevant
- Merging the News Desk, Section Name, SubSectionName into one variable as they seem to have a unique relationship