Predicting When Articles Go Viral

Shristi Singh

March 12, 2020

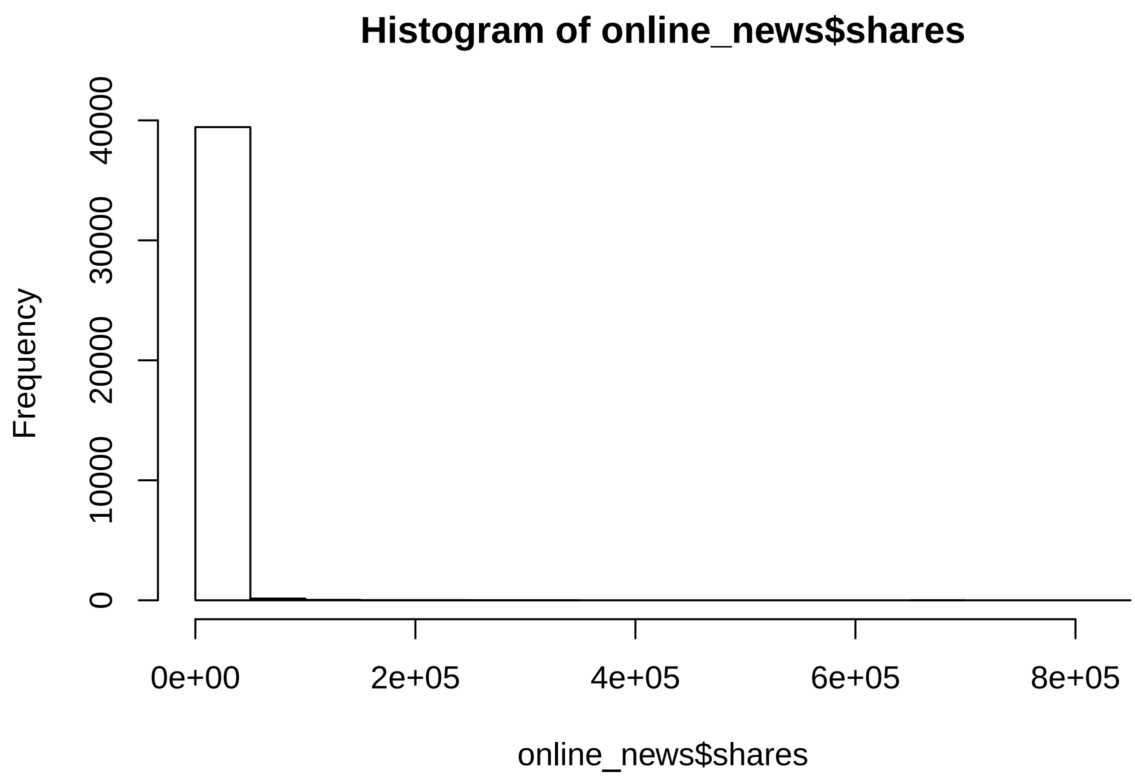
```
##### Importiing, viewing, and analizing data
online_news <- read.csv("~/GitHub/SDS323_Spring2020/hw2/q3/online_news.csv")
View(online_news)
str(online_news)
```

```
## 'data.frame':    39644 obs. of  38 variables:
##  $ url                          : Factor w/ 39644 levels "http://mashable.com/2013/01/07/amazon-insta
##  $ n_tokens_title               : int  12 9 9 9 13 10 8 12 11 10 ...
##  $ n_tokens_content             : int  219 255 211 531 1072 370 960 989 97 231 ...
##  $ num_hrefs                    : int  4 3 3 9 19 2 21 20 2 4 ...
##  $ num_self_hrefs               : int  2 1 1 0 19 2 20 20 0 1 ...
##  $ num_imgs                     : int  1 1 1 1 20 0 20 20 0 1 ...
##  $ num_videos                   : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ average_token_length         : num  4.68 4.91 4.39 4.4 4.68 ...
##  $ num_keywords                 : int  5 4 6 7 7 9 10 9 7 5 ...
##  $ data_channel_is_lifestyle    : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ data_channel_is_entertainment: int  1 0 0 1 0 0 0 0 0 0 ...
##  $ data_channel_is_bus          : int  0 1 1 0 0 0 0 0 0 0 ...
##  $ data_channel_is_socmed       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ data_channel_is_tech         : int  0 0 0 0 1 1 0 1 1 0 ...
##  $ data_channel_is_world        : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ self_reference_min_shares    : num  496 0 918 0 545 8500 545 545 0 0 ...
##  $ self_reference_max_shares    : num  496 0 918 0 16000 8500 16000 16000 0 0 ...
##  $ self_reference_avg_sharess   : num  496 0 918 0 3151 ...
##  $ weekday_is_monday            : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ weekday_is_tuesday           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_wednesday         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_thursday          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_friday            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_saturday          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday_is_sunday            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ is_weekend                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ global_rate_positive_words   : num  0.0457 0.0431 0.0569 0.0414 0.0746 ...
##  $ global_rate_negative_words   : num  0.0137 0.01569 0.00948 0.02072 0.01213 ...
##  $ avg_positive_polarity        : num  0.379 0.287 0.496 0.386 0.411 ...
##  $ min_positive_polarity        : num  0.1 0.0333 0.1 0.1364 0.0333 ...
##  $ max_positive_polarity        : num  0.7 0.7 1 0.8 1 0.6 1 1 0.8 0.5 ...
##  $ avg_negative_polarity        : num  -0.35 -0.119 -0.467 -0.37 -0.22 ...
##  $ min_negative_polarity        : num  -0.6 -0.125 -0.8 -0.6 -0.5 -0.4 -0.5 -0.5 -0.125 -0.5 ...
##  $ max_negative_polarity        : num  -0.2 -0.1 -0.133 -0.167 -0.05 ...
##  $ title_subjectivity           : num  0.5 0 0 0 0.455 ...
##  $ title_sentiment_polarity     : num  -0.188 0 0 0 0.136 ...
##  $ abs_title_sentiment_polarity : num  0.188 0 0 0 0.136 ...
```

```
##  $ shares                        : int  593 711 1500 1200 505 855 556 891 3600 710 ...
```

```
##### Model 1 Regress first and threshold second

hist(online_news$shares)
```



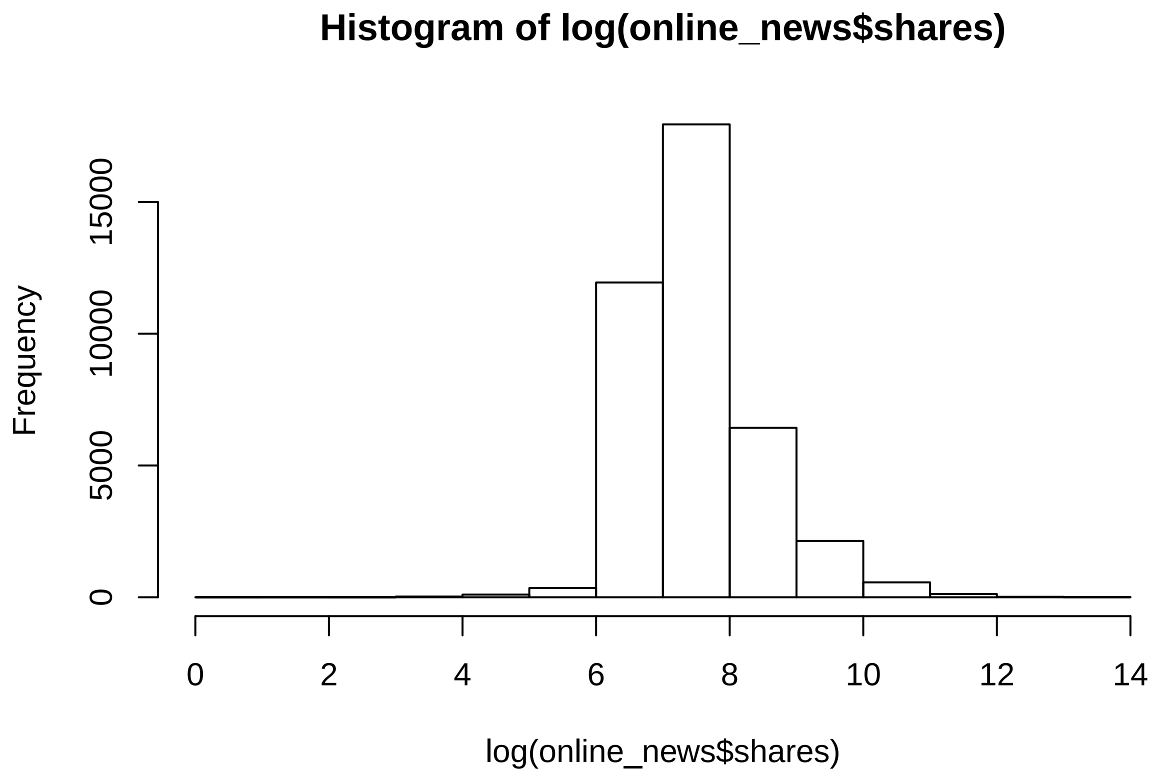

```
# We should apply the log transformation since shares is very skewed

# After log transformation
hist(log(online_news$shares))

# Fitting lasso regression and doing cross validation of K=10 folds to automate finding independent var

library(gamlr)
```

```
## Warning: package 'gamlr' was built under R version 3.6.3
## Loading required package: Matrix
```



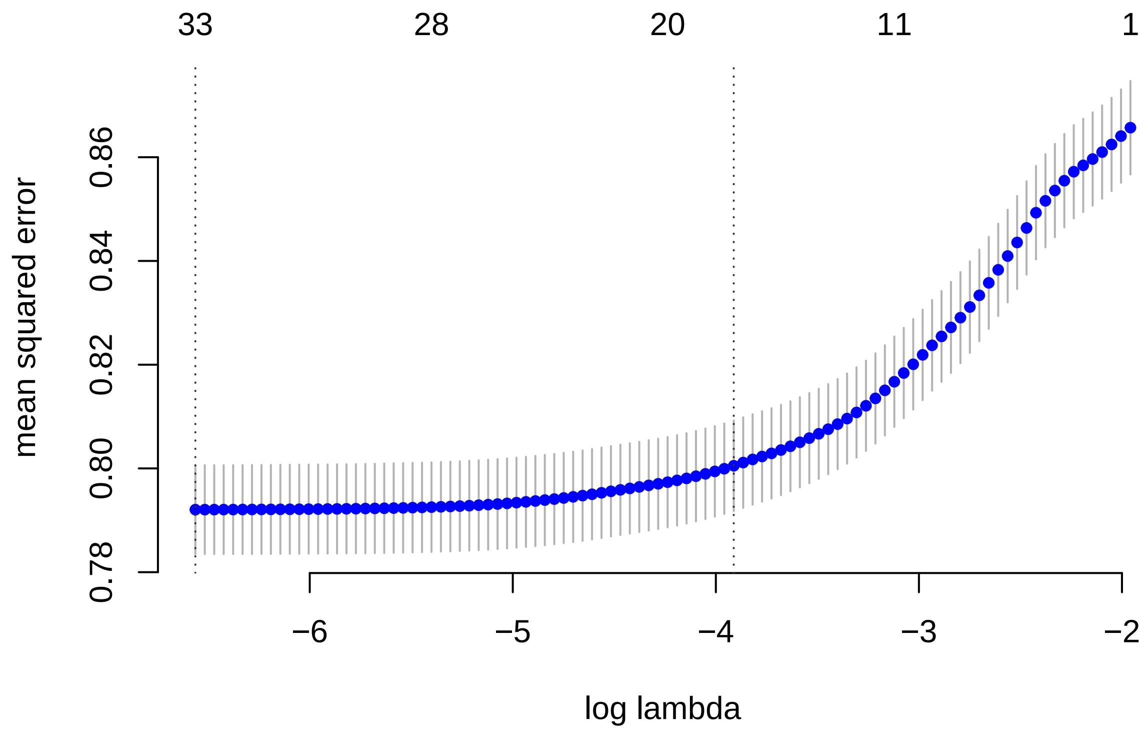

```r
# Creating a matrix of all the independent varaibles exculuding url from online_news data using the spa
x = sparse.model.matrix(log(shares) ~ . - url, data=online_news)[,-1] # -1 drops intercept

y = log(online_news$shares) # Pulling out `y' for convenience and taking the log of the dependent varia

# Fiting lasso regression to the data and doing cross validation of k=10 folds using the cv.gamlr comma
# Verb = TRUE prints progress
cvl = cv.gamlr(x, y, nfold=10, verb=TRUE)
```

```
## fold 1,2,3,4,5,6,7,8,9,10,done.
```

```r
# Plotting out-of-sample deviance as a function of log lambda
plot(cvl, bty="n")
```




```
## CV minimum deviance selection
b.min = coef(cvl, select="min")
# value of lamda:
log(cvl$lambda.min)
```

```
## [1] -6.563407
```

```
sum(b.min!=0) # this gives the coefficent not 0
```

```
## [1] 33
```

```
##########

# Predict number of shares
lhat_shares = predict(cvl, x) # log value of shares
hat_shares = exp(lhat_shares) # predicted values of shares
head (hat_shares, 50)
```

```
## 50 x 1 Matrix of class "dgeMatrix"
##       seg43
## 1  1401.945
## 2  1534.302
## 3  1645.311
## 4  1455.210
## 5  2083.167
## 6  1898.552
## 7  2123.333
## 8  2205.976
```

```
## 9  1754.700
## 10 1239.605
## 11 1360.298
## 12 1837.373
## 13 2250.904
## 14 2008.537
## 15 2064.572
## 16 1337.521
## 17 2096.992
## 18 1596.555
## 19 1850.922
## 20 2291.425
## 21 2119.145
## 22 1267.083
## 23 2064.195
## 24 1298.560
## 25 1667.425
## 26 1943.174
## 27 1846.313
## 28 2258.036
## 29 1824.547
## 30 1743.791
## 31 1667.216
## 32 1897.006
## 33 1913.750
## 34 1972.902
## 35 1872.446
## 36 1963.086
## 37 2273.345
## 38 2135.496
## 39 1288.238
## 40 1381.168
## 41 2096.617
## 42 1737.770
## 43 2253.163
## 44 2141.645
## 45 2071.146
## 46 1602.422
## 47 2127.236
## 48 1518.198
## 49 2100.655
## 50 2076.456
```

```r
# Changing predicted number of shares into viral prediction(t_viral)
threshold_viral = ifelse(hat_shares > 1400, 1, 0)
head(threshold_viral, 50)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 0 0 1 1 1 1 1 1 1 1 1 1
```

```r
# Creating new variable "viral"
viral = ifelse(online_news$shares > 1400, 1, 0)
head(viral, 20)
```

```
##  [1] 0 0 1 0 0 0 0 0 1 0 1 1 0 1 0 1 1 1 1 1
```

```
# Creating confusion matrix
confusion_1= table(y = viral, yhat = threshold_viral)
print(confusion_1)
```

```
##    yhat
## y       0     1
##   0  4688 15394
##   1  1953 17609
```

```
sum(diag(confusion_1))/sum(confusion_1) # This gives the sample accuracy for model 1
```

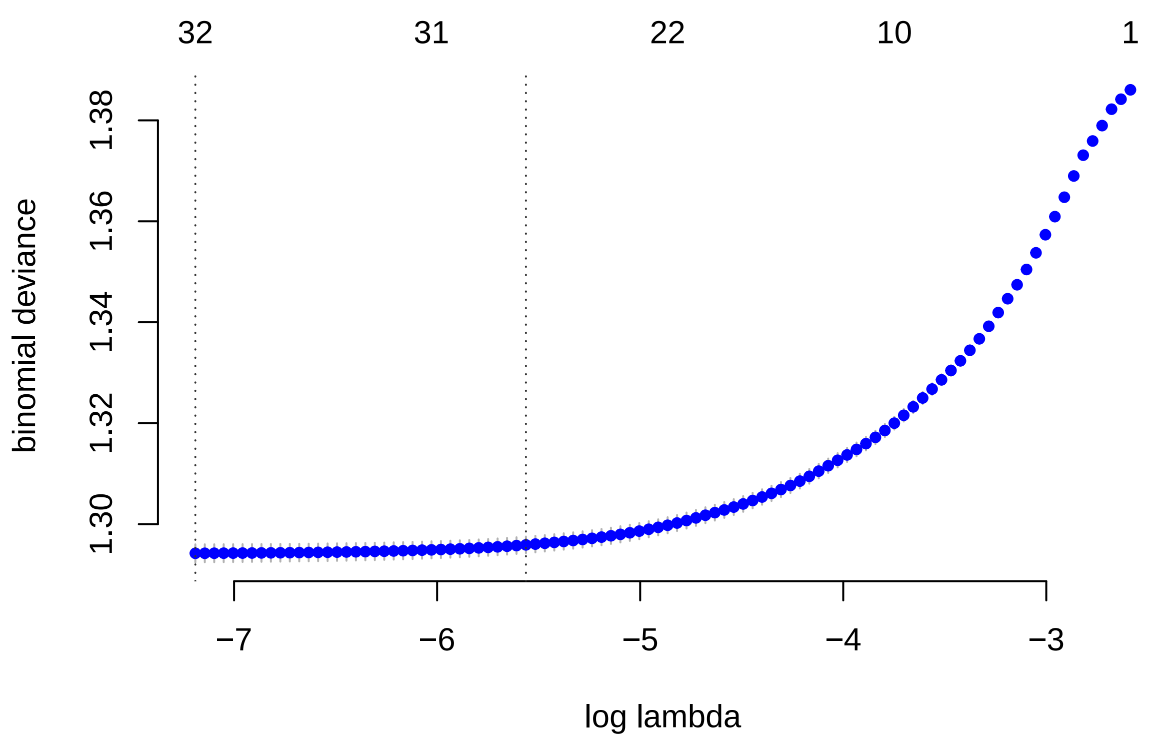
```
## [1] 0.5624306
```

```
##### Model 2 Threshold first and regress/classify second.

# Running logistic lasso regression and cross validate with viral as the dependent variable
# family = "binomial" in this code is used to do a logistic regression instead of normal regression
#(verb just prints progress)
viral_cvl = cv.gamlr(x, viral, nfold=10, family="binomial", verb=TRUE)
```

```
## fold 1,2,3,4,5,6,7,8,9,10,done.
```

```
# Plotting  the out-of-sample deviance as a function of log lambda
plot(viral_cvl, bty="n")
```




```
## CV minimum deviance selection
b.min = coef(viral_cvl, select="min")
log(viral_cvl$lambda.min)
```

```
## [1] -7.190693
```

```r
sum(b.min!=0) # This is random because of the CV randomness.
```

```
## [1] 32
```

```r
# Predicting number of viral
hat_viral = predict(viral_cvl, x)
head (hat_viral, 50)
```

```
## 50 x 1 Matrix of class "dgeMatrix"
##          seg65
## 1  -0.83753266
## 2  -0.33612009
## 3  -0.18654960
## 4  -0.63862338
## 5   0.27051597
## 6   0.29104341
## 7   0.18588451
## 8   0.43840567
## 9  -0.01515228
## 10 -0.96462771
## 11 -0.60422856
## 12 -0.02553491
## 13  0.24505945
## 14 -0.05795172
## 15  0.28780215
## 16 -0.73688086
## 17  0.09923332
## 18 -0.16204395
## 19 -0.13624867
## 20  0.39719746
## 21  0.34033879
## 22 -0.85803305
## 23  0.20832478
## 24 -0.75981551
## 25 -0.33506149
## 26  0.32648834
## 27  0.20173882
## 28  0.45006171
## 29  0.04588509
## 30  0.06015864
## 31 -0.08565285
## 32  0.18299222
## 33  0.26884003
## 34  0.29559418
## 35  0.05835040
## 36  0.18308295
## 37  0.37201983
## 38  0.64332181
## 39 -0.87508241
## 40 -0.76958635
## 41  0.27658767
## 42  0.06658661
## 43  0.29677771
```

```
## 44  0.38878585
## 45  0.23408925
## 46 -0.31363037
## 47  0.11646748
## 48 -0.53475500
## 49  0.27302751
## 50  0.23502487
```

```r
# Changing hat_viral to true/false prediction
b_hat_viral = ifelse(hat_viral > 0, 1, 0)
head(b_hat_viral, 50)
```

```
##  [1] 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1
## [39] 0 0 1 1 1 1 1 0 1 0 1 1
```

```r
# Creating confusion matirx
confusion_2= table(y = viral, yhat = b_hat_viral)
print(confusion_2)
```

```
##    yhat
## y       0     1
##   0 12353  7729
##   1  6937 12625
```

```r
sum(diag(confusion_2))/sum(confusion_2) # This is the sample accuracy of model 2
```

```
## [1] 0.6300575
```

```r
##### Comaprison of models

table(viral) # The actual number of viral or not viral articles
```

```
## viral
##     0     1
## 20082 19562
```

```r
20082/39644  # 50.66 percent of articles were not viral which is the null hypothesis
```

```
## [1] 0.5065584
```

```r
print(confusion_1)
```

```
##    yhat
## y       0     1
##   0  4688 15394
##   1  1953 17609
```

```r
sum(diag(confusion_1))/sum(confusion_1) # The sample accuracy for model 1 is 56.8 percent
```

```
## [1] 0.5624306
```

```r
# Hence model 1 is (56.8-50.66) about a 6 percent improvement to the null model
17458/(17458+5058) # True positive rate of model 1 is 77.54 percent
```

```
## [1] 0.7753597
```

```r
15024/(5058+15024) # Fasle positive rate of model 1 is 74.81 percent
```

```
## [1] 0.7481327
```

```r
15024/(15024+17458)# False dicovery rate of model 1 is 46.25 percent
```

```
## [1] 0.4625331
```

```r
print(confusion_2)
```

```
##    yhat
## y       0     1
##   0 12353  7729
##   1  6937 12625
```

```r
sum(diag(confusion_2))/sum(confusion_2) # The sample accuracy of model 2 is 63 percent
```

```
## [1] 0.6300575
```

```r
# Hence model 2 is 12.5 percent improvement to null model and about 6.2 percent improvement to model 1
12704/(12705+6857) # True positive rate is 64.95 percent which is worst than model 1
```

```
## [1] 0.6494223
```

```r
7811/(7811+12271) # False positive rate is 38.9 percent which is better than model 1 because lower is b
```

```
## [1] 0.3889553
```

```r
7811/(7811+12705) # False discovery rate is 38.07 percent which is better than model 1 because lower is
```

```
## [1] 0.3807272
```

```r
# In conclusion based on True Positive Rate, False Positve Rate, False Discovery Rate, and general acur
```