## News Popularity Dataset

We load the dataset and select the required variables only.

newsShort <- read\_csv("OnlineNewsPopularity.csv")%>%  
 select("n\_tokens\_title", "n\_tokens\_content", "n\_unique\_tokens", "n\_non\_stop\_words", "num\_hrefs", "num\_imgs", "num\_videos", "average\_token\_length", "num\_keywords", "kw\_max\_max", "global\_sentiment\_polarity", "avg\_positive\_polarity", "title\_subjectivity", "title\_sentiment\_polarity", "abs\_title\_subjectivity", "abs\_title\_sentiment\_polarity", "shares")

### Data Pre-Processing

Create a new variable called popular if the share is higher than 1400, 1400 is the median of the shares.

newsShort <- newsShort%>%  
 mutate(popular=if\_else((shares >= 1400),1,0))%>%  
 select(-shares)  
newsShort$popular <- as.factor(newsShort$popular)  
glimpse(newsShort)

## Observations: 39,644  
## Variables: 17  
## $ n\_tokens\_title <dbl> 12, 9, 9, 9, 13, 10, 8, 12, 11, 1...  
## $ n\_tokens\_content <dbl> 219, 255, 211, 531, 1072, 370, 96...  
## $ n\_unique\_tokens <dbl> 0.6635945, 0.6047431, 0.5751295, ...  
## $ n\_non\_stop\_words <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...  
## $ num\_hrefs <dbl> 4, 3, 3, 9, 19, 2, 21, 20, 2, 4, ...  
## $ num\_imgs <dbl> 1, 1, 1, 1, 20, 0, 20, 20, 0, 1, ...  
## $ num\_videos <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ...  
## $ average\_token\_length <dbl> 4.680365, 4.913725, 4.393365, 4.4...  
## $ num\_keywords <dbl> 5, 4, 6, 7, 7, 9, 10, 9, 7, 5, 8,...  
## $ kw\_max\_max <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...  
## $ global\_sentiment\_polarity <dbl> 0.09256198, 0.14894781, 0.3233333...  
## $ avg\_positive\_polarity <dbl> 0.3786364, 0.2869146, 0.4958333, ...  
## $ title\_subjectivity <dbl> 0.5000000, 0.0000000, 0.0000000, ...  
## $ title\_sentiment\_polarity <dbl> -0.1875000, 0.0000000, 0.0000000,...  
## $ abs\_title\_subjectivity <dbl> 0.00000000, 0.50000000, 0.5000000...  
## $ abs\_title\_sentiment\_polarity <dbl> 0.1875000, 0.0000000, 0.0000000, ...  
## $ popular <fct> 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, ...

news\_rand <- newsShort[order(runif(10000)), ]  
set.seed(12345)

#Split the data into training and test datasets  
news\_train <- news\_rand[1:9000, ]  
news\_test <- news\_rand[9001:10000, ]

### Full Model

nb\_model <- naive\_bayes(popular ~ ., data=news\_train)  
nb\_model

## ================================ Naive Bayes =================================   
## Call:   
## naive\_bayes.formula(formula = popular ~ ., data = news\_train)  
##   
## A priori probabilities:   
##   
## 0 1   
## 0.4291111 0.5708889   
##   
## Tables:   
##   
## n\_tokens\_title 0 1  
## mean 9.820559 9.695991  
## sd 1.929249 1.987754  
##   
##   
## n\_tokens\_content 0 1  
## mean 452.2315 515.1051  
## sd 347.1779 450.0206  
##   
##   
## n\_unique\_tokens 0 1  
## mean 0.5702437 0.5542023  
## sd 0.1127776 0.1232687  
##   
##   
## n\_non\_stop\_words 0 1  
## mean 0.99404453 0.99124172  
## sd 0.07695147 0.09318398  
##   
##   
## num\_hrefs 0 1  
## mean 9.147851 10.570650  
## sd 8.644083 11.540711  
##   
## # ... and 11 more tables

### Create Prediction

news\_Pred <- predict(nb\_model, newdata = news\_test)  
conf\_nat <- table(news\_Pred, news\_test$popular)  
conf\_nat

##   
## news\_Pred 0 1  
## 0 329 400  
## 1 101 170

Accuracy <- sum(diag(conf\_nat))/sum(conf\_nat)\*100  
Accuracy

## [1] 49.9

Not great.

### Optimization

To optimize the model, we will look how we can remove variables which are correlated with each other and remove the highly correlated ones withoput affecting the model.

newsDataScaled <- scale(news\_rand[,0:(ncol(news\_rand)-1)], center=TRUE, scale = TRUE)  
m <- cor(newsDataScaled)  
highlycor <- findCorrelation(m, 0.30)  
highlycor

## [1] 3 16 2 4 12 13

These are the indices of the variables that are highly correlated with each other. Below, we run a correlation of these variables with the dependent variables.

#check how the above variables are correlated with the dependent variable  
check <- news\_rand%>%select(3,16,2,4,12,13,17)  
check$popular<-as.numeric(check$popular)  
cor(check)

## n\_unique\_tokens abs\_title\_sentiment\_polarity  
## n\_unique\_tokens 1.000000000 -0.004251095  
## abs\_title\_sentiment\_polarity -0.004251095 1.000000000  
## n\_tokens\_content -0.626662198 0.011187585  
## n\_non\_stop\_words 0.417037438 -0.023983298  
## avg\_positive\_polarity 0.154500778 0.140369050  
## title\_subjectivity 0.025234677 0.725455997  
## popular -0.062992819 0.026636966  
## n\_tokens\_content n\_non\_stop\_words  
## n\_unique\_tokens -0.626662198 0.41703744  
## abs\_title\_sentiment\_polarity 0.011187585 -0.02398330  
## n\_tokens\_content 1.000000000 0.10552159  
## n\_non\_stop\_words 0.105521587 1.00000000  
## avg\_positive\_polarity 0.078679331 0.34982582  
## title\_subjectivity -0.009127765 -0.03605004  
## popular 0.067211377 -0.01946947  
## avg\_positive\_polarity title\_subjectivity  
## n\_unique\_tokens 0.154500778 0.025234677  
## abs\_title\_sentiment\_polarity 0.140369050 0.725455997  
## n\_tokens\_content 0.078679331 -0.009127765  
## n\_non\_stop\_words 0.349825816 -0.036050041  
## avg\_positive\_polarity 1.000000000 0.081716910  
## title\_subjectivity 0.081716910 1.000000000  
## popular 0.008526717 0.018061939  
## popular  
## n\_unique\_tokens -0.062992819  
## abs\_title\_sentiment\_polarity 0.026636966  
## n\_tokens\_content 0.067211377  
## n\_non\_stop\_words -0.019469471  
## avg\_positive\_polarity 0.008526717  
## title\_subjectivity 0.018061939  
## popular 1.000000000

findCorrelation(m,0.6)

## [1] 3 16 4

Below, we create a filtered dataset by disselecting the varaibles that are highly likely to create high pairwise correlation, applied trial & error basis.

filteredData <- news\_rand%>%select(-n\_unique\_tokens,-n\_non\_stop\_words,-abs\_title\_sentiment\_polarity,-num\_keywords)  
filteredTraining <- filteredData[1:750, ]  
filteredTest <- filteredData[751:1000, ]

### 

### Optimized Model

Training the Model:

nb\_model <- naive\_bayes(popular ~ ., data=filteredTraining)  
nb\_model

## ================================ Naive Bayes =================================   
## Call:   
## naive\_bayes.formula(formula = popular ~ ., data = filteredTraining)  
##   
## A priori probabilities:   
##   
## 0 1   
## 0.4173333 0.5826667   
##   
## Tables:   
##   
## n\_tokens\_title 0 1  
## mean 9.616613 9.704805  
## sd 1.903096 2.013099  
##   
##   
## n\_tokens\_content 0 1  
## mean 475.8658 524.9130  
## sd 359.3243 447.9918  
##   
##   
## num\_hrefs 0 1  
## mean 9.092652 11.226545  
## sd 8.875260 11.738411  
##   
##   
## num\_imgs 0 1  
## mean 3.539936 3.951945  
## sd 6.977471 8.279242  
##   
##   
## num\_videos 0 1  
## mean 1.300319 1.212815  
## sd 5.538261 4.883789  
##   
## # ... and 7 more tables

Evaluating the Model:

filteredTestPred <- predict(nb\_model, newdata = filteredTest)  
table(filteredTestPred, filteredTest$popular)

##   
## filteredTestPred 0 1  
## 0 86 83  
## 1 26 55

Creating Confusion Matrix:

tab <- table(filteredTestPred, filteredTest$popular)  
caret::confusionMatrix(tab)

## Confusion Matrix and Statistics  
##   
##   
## filteredTestPred 0 1  
## 0 86 83  
## 1 26 55  
##   
## Accuracy : 0.564   
## 95% CI : (0.5001, 0.6264)  
## No Information Rate : 0.552   
## P-Value [Acc > NIR] : 0.3761   
##   
## Kappa : 0.1588   
## Mcnemar's Test P-Value : 8.148e-08   
##   
## Sensitivity : 0.7679   
## Specificity : 0.3986   
## Pos Pred Value : 0.5089   
## Neg Pred Value : 0.6790   
## Prevalence : 0.4480   
## Detection Rate : 0.3440   
## Detection Prevalence : 0.6760   
## Balanced Accuracy : 0.5832   
##   
## 'Positive' Class : 0   
##

Calculating Model Accuracy:

conf\_nat <- table(filteredTestPred, filteredTest$popular)  
conf\_nat

##   
## filteredTestPred 0 1  
## 0 86 83  
## 1 26 55

Accuracy <- sum(diag(conf\_nat))/sum(conf\_nat)\*100  
Accuracy

## [1] 56.4

Accuracy is better.

### Results & Discussion

We are trying to predict the popularity of the news by using the Naïve Bayes Model. For this assignment, we created a full model with all the variables and the accuracy was only 49.9.

To further better that model, we selected variables based on the correlation between them. Selection was made by taking out highly correlated variables that did not affect the dependent variable. A 56.4 accuracy was achieved with that approach.