1. **Find an interesting dataset that is appropriate for applying the naive Bayes algorithm and try to load the data into R and proceed to classify the data using naive Bayes.**

**Twitter dataset**

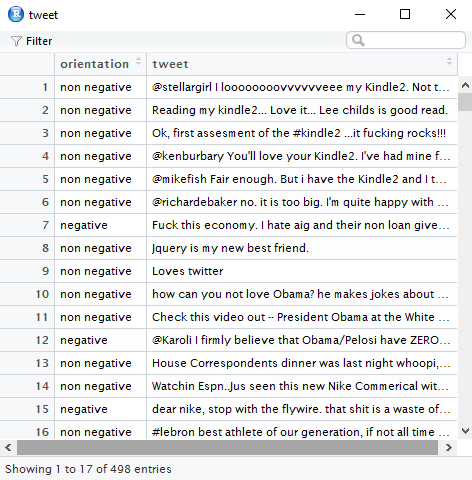
The twitter dataset is collected from the web and it contains 498 observations and 3 categorical variables.

The main goal of this dataset is to classify the difference between negative and non-negative terms using naïve bayes algorithm.

1. **Collecting the data**

* We are going to the read the dataset(twitter) that we collected from the web which contains 498 observations and 3 categorical variables(negative,non-negative,tweet)
* We are going to construct the structure to see the exact dataframe.

> twitter <- read.csv("twitter.csv", stringsAsFactors=FALSE)



1. **Exploring the data**

> str(twitter)

'data.frame': 498 obs. of 2 variables:

$ orientation: chr "non negative" "non negative" "non negative"

"non negative" ...

$ tweet : chr "@stellargirl I loooooooovvvvvveee my Kindle2.

Not that the DX is cool, but the 2 is fantastic in its own right." "

Reading my kindle2... Love it... Lee childs is good read." "Ok, fir

st assesment of the #kindle2 ...it fucking rocks!!!" "@kenburbary Yo

u'll love your Kindle2. I've had mine for a few months and never loo

ked back. The new big one is huge! No need fo"| \_\_truncated\_\_ ...

* We are going to factorize the dataset based on the tweet and tabulate the result in the form of negative and non-negative.

twitter$tweet <- factor(twitter$tweet)

> str(twitter$tweet)

Factor w/ 498 levels "' Barack Obama shows his funny side \" &gt;&g

t; http://tr.im/l0gY !! Great speech..",..: 86 385 365 45 57 70 176

281 316 208 ...

> table(twitter$tweet)

' Barack Obama shows his funny side " &gt;&gt; http://tr.im/l0gY !!

Great speech..

'Next time, I'll call myself Nike'

* We are going to create a corpus using text mining package

> twitter$tweet <- factor(twitter$tweet)

> library(tm)

> twitter\_corpus <- Corpus(VectorSource(twitter$tweet))

> print(twitter\_corpus)

<<VCorpus>>

Metadata: corpus specific: 0, document level (indexed): 0

Content: documents: 498

* Clean the corpus using tm\_map function to remove words,numbers,punctuations.
* Examine the clean corpus.

> corpus\_clean<-tm\_map(twitter\_corpus, tolower)

> corpus\_clean<-tm\_map(corpus\_clean, removeNumbers)

> corpus\_clean<-tm\_map(corpus\_clean, removeWords, stopwords())

> corpus\_clean<-tm\_map(corpus\_clean, removePunctuation)

> corpus\_clean<-tm\_map(corpus\_clean, stripWhitespace)

> corpus\_clean <- tm\_map(corpus\_clean, PlainTextDocument)

> #examine the clean corpus

> lapply(twitter\_corpus[1:3], as.character)

$`1`

[1] "@stellargirl I loooooooovvvvvveee my Kindle2. Not that the DX i

s cool, but the 2 is fantastic in its own right."

$`2`

[1] "Reading my kindle2... Love it... Lee childs is good read."

$`3`

[1] "Ok, first assesment of the #kindle2 ...it fucking rocks!!!"

> lapply(corpus\_clean[1:3], as.character)

$`character(0)`

[1] "stellargirl loooooooovvvvvveee kindle dx cool fantastic right"

$`character(0)`

[1] "reading kindle love lee childs good read"

$`character(0)`

[1] "ok first assesment kindle fucking rocks"

* Creating the document term sparse matrix which involves in a single command.

> twitter\_dtm <- DocumentTermMatrix(corpus\_clean)

> twitter\_dtm

<<DocumentTermMatrix (documents: 498, terms: 2036)>>

Non-/sparse entries: 3930/1009998

Sparsity : 100%

Maximal term length: 46

Weighting : term frequency (tf)

1. **Creating training and test datasets**

* We are going to create training and test datasets and applying wordcloud() function to train and test data sets for highlighting most frequently appeared words.

> twitter\_raw\_train <- twitter[1:300,]

> twitter\_raw\_test <- twitter[301:498,]

> twitter\_dtm\_train <- twitter\_dtm[1:300, ]

> twitter\_dtm\_test <- twitter\_dtm[301:498, ]

> twitter\_corpus\_train <- corpus\_clean[1:300]

> twitter\_corpus\_test <- corpus\_clean[301:498]

> # check that the proportion of spam is similar

> prop.table(table(twitter\_raw\_train$type))

numeric(0)

> prop.table(table(twitter\_raw\_test$type))

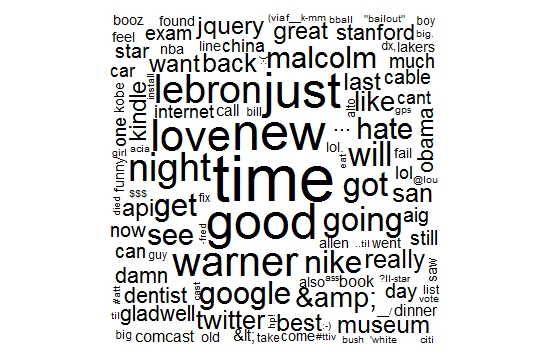
numeric(0)

> # word cloud visualization

> library(wordcloud)

> wordcloud(twitter\_corpus\_train, min.freq = 30, random.order = FALSE

)



* Subseting the training as we did earlier based on the orientation and applying wordcloud to find the difference between negative and non negative words.

> non\_negative <- subset(tweet\_raw\_train, orientation == "non

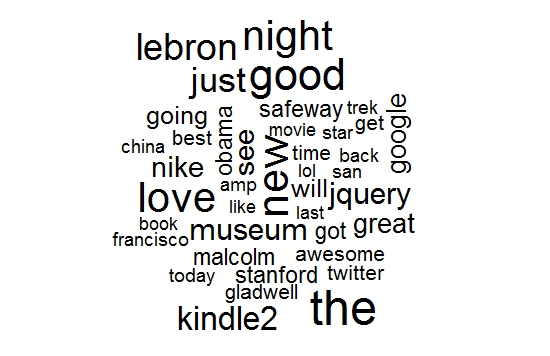
negative")

> negative <- subset(tweet\_raw\_train, orientation == "negative

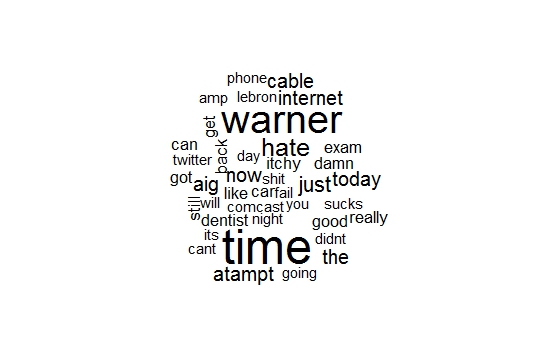
")

> wordcloud(non\_negative$tweet, max.words = 40, scale = c(3, 0

.5))

****

> wordcloud(negative$tweet, max.words = 40, scale = c(3, 0.5))

****

* We are going to find the frequency words we used in the dataset by using findfreq() function.
* Converting the counts to a factor.

|  |
| --- |
| > tweet\_dict <- c(findFreqTerms(tweet\_dtm\_train, 5))  > tweet\_train <- DocumentTermMatrix(tweet\_corpus\_train,  list(dictionary = tweet\_dict))  > tweet\_test <- DocumentTermMatrix(tweet\_corpus\_test,  list(dictionary = tweet\_dict))  > # convert counts to a factor  > convert\_counts <- function(x) {  + x <- ifelse(x > 0, 1, 0)  + x <- factor(x, levels = c(0, 1), labels = c("negative", "non negative"))  + } |
| * Applying converts to columns of train and test data.   > tweet\_train <- apply(tweet\_train, MARGIN = 2, convert\_counts)  > tweet\_test <- apply(tweet\_test, MARGIN = 2, convert\_counts)  > summary(tweet\_train[, 1:5])   * Training the model by applying naivebayes algorithm   > library(e1071)  > tweet\_classifier <- naiveBayes(tweet\_train, tweet\_raw\_train$orientation)  > names(tweet\_classifier)  [1] "apriori" "tables" "levels" "call"  > tweet\_classifier$tables[1:2]  $negative  negative  tweet\_raw\_train$orientation non negative  negative 1  non negative 1  $non  non  tweet\_raw\_train$orientation negative non negative  negative 1 0  non negative 0 1  > tweet\_classifier  Naive Bayes Classifier for Discrete Predictors  Call:  naiveBayes.default(x = tweet\_train, y = tweet\_raw\_train$orientation)  A-priori probabilities:  tweet\_raw\_train$orientation  negative non negative  0.3566667 0.6433333  Conditional probabilities:  negative  tweet\_raw\_train$orientation non negative  negative 1  non negative 1  non  tweet\_raw\_train$orientation negative non negative  negative 1 0  non negative 0 1  > tweet\_classifier <- naiveBayes(tweet\_train, tweet\_raw\_train$tweet)  > names(tweet\_classifier)  [1] "apriori" "tables" "levels" "call"  > tweet\_classifier$tables[1:2]  $negative    negative  tweet\_raw\_train$tweet  non negative  ' Barack Obama shows his funny side " &gt;&gt; http://tr.im/l0gY !!  Great speech..  1  'Next time, I'll call myself Nike'   1. **Evaluating the model performance**  * We are going to evaluate the model performance based on the orientation and applying cross tables.   > tweet\_test\_pred <- predict(tweet\_classifier, tweet\_test)  > library(gmodels)  > CrossTable(tweet\_test\_pred, tweet\_raw\_test$orientation,  + prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,  + dnn = c('predicted', 'actual'))      Cell Contents  |-------------------------|  | N |  | N / Col Total |  |-------------------------|    Total Observations in Table: 198    | actual  predicted | negative | non negative | Row Total |  -------------|--------------|--------------|--------------|  negative | 70 | 0 | 70 |  | 1.000 | 0.000 | |  -------------|--------------|--------------|--------------|  non negative | 0 | 128 | 128 |  | 0.000 | 1.000 | |  -------------|--------------|--------------|--------------|  Column Total | 70 | 128 | 198 |  | 0.354 | 0.646 | |  -------------|--------------|--------------|--------------| |
| 1. **Improving the model performance**  * We are going to build a Naive Bayes model as done earlier, but this time set laplace = 1 and laplace = 2.   > tweet\_classifier2 <- naiveBayes(tweet\_train, tweet\_raw\_train$orientation,  laplace = 1)  > tweet\_test\_pred2 <- predict(tweet\_classifier2, tweet\_test)  > CrossTable(tweet\_test\_pred2, tweet\_raw\_test$orientation,  + prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,  + dnn = c('predicted', 'actual'))    Cell Contents  |-------------------------|  | N |  | N / Col Total |  |-------------------------|    Total Observations in Table: 198    | actual  predicted | negative | non negative | Row Total |  -------------|--------------|--------------|--------------|  negative | 70 | 0 | 70 |  | 1.000 | 0.000 | |  -------------|--------------|--------------|--------------|  non negative | 0 | 128 | 128 |  | 0.000 | 1.000 | |  -------------|--------------|--------------|--------------|  Column Total | 70 | 128 | 198 |  | 0.354 | 0.646 | |    > tweet\_classifier3 <- naiveBayes(tweet\_train, tweet\_raw\_train$orientation,  laplace = 2)  > tweet\_test\_pred3 <- predict(tweet\_classifier3, tweet\_test)  > CrossTable(tweet\_test\_pred3, tweet\_raw\_test$orientation,  + prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,  + dnn = c('predicted', 'actual'))    Cell Contents  |-------------------------|  | N |  | N / Col Total |  |-------------------------|    Total Observations in Table: 198    | actual  predicted | negative | non negative | Row Total |  -------------|--------------|--------------|--------------|  negative | 70 | 0 | 70 |  | 1.000 | 0.000 | |  -------------|--------------|--------------|--------------|  non negative | 0 | 128 | 128 |  | 0.000 | 1.000 | |  -------------|--------------|--------------|--------------|  Column Total | 70 | 128 | 198 |  | 0.354 | 0.646 | |  -------------|--------------|--------------|--------------| |
| |  | | --- | | **Conclusion** |  * The model with laplace 1 and laplace 2 has 100% accuracy with 70 True positive negative tweets and   128 true positive non-negative tweets. |
|  |
|  |