# SUPERVISED LEARNING METHOD:

# NAÏVE BAYES ALGORITHM: SMS SPAM DATA

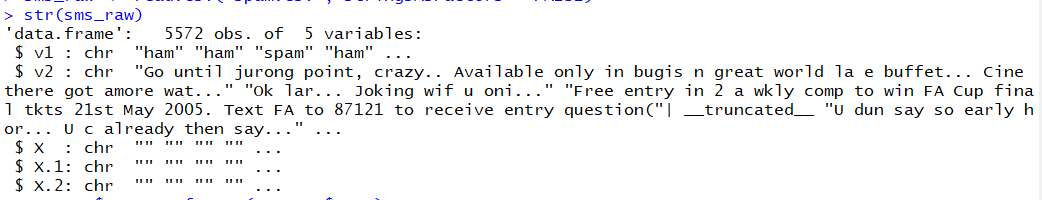
**Naïve Bayes** is an interesting application of the Bayes theorem to the problem of classification. Many machine learning methods use Bayesian Probability, but Naïve Bayes is the most common.

## Data Introduction:

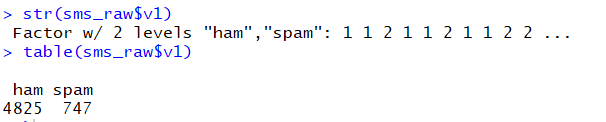
The sms spam data was sourced from UCI Machine Learning Repository. The dataset has 5572 observations that belong to two different categories spam(unwanted messages) or ham(desired messages). By looking at the data and checking for various patterns, we can form probabilities that a message is spam or ham based on the words within the texts.

## Data Exploration:

The data has two columns v1 and v2. V1 represents the type of type of email and v2 represents the text message.



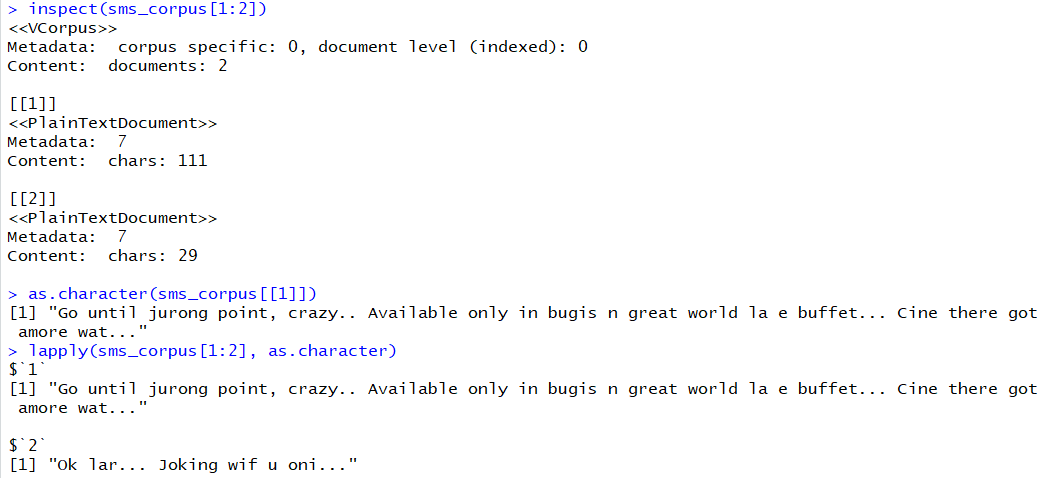
As we can see that, type has two levels. There are 4825 messages in ham category and 747 in spam.

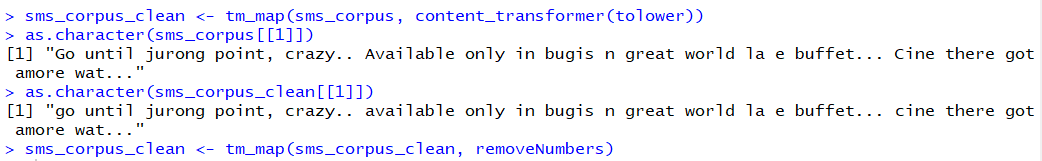


## Data Cleaning:

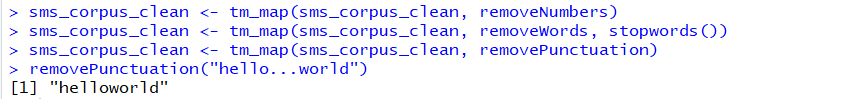
I will use R package tm to clean the strings in the data. This package has a function Vcorpus() which creates a corpus, or a body of text documents as a list object.

Inspect() will give individual sms from our new list sms\_corpus.

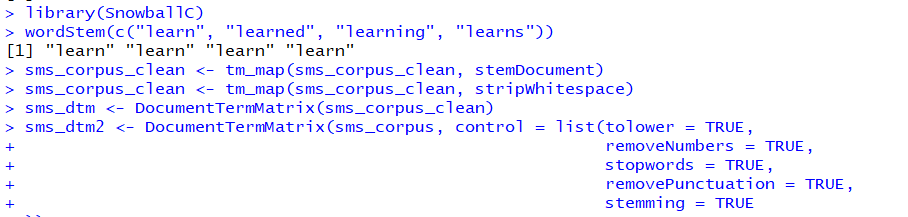




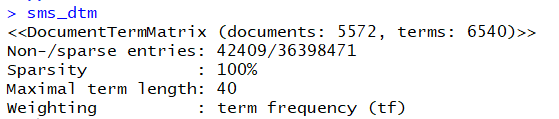
I will next standardize the documents within our corpus to ensure that the computer can properly characterize them. Strings like **Hello!,** **Hello** and **hello** will be modified and stop words like **to, and, but, or** will be removed. After this, I will stem the documents. Stemming refers to the process where a word is stripped to its root word. For example, a verb with suffixes es, ed, ing will be removed. For this, I will use snowball package. I will also remove the whitespaces in the documents.

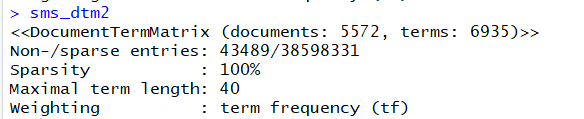


The final step in the standardization process is to split the messages into individual words while still maintaining the identity of the messages. This can be done through a process called **tokenization**, where each word in the text string is considered a token. To do so, I will use the function DocumentTermMatrix() This function splits a single text message into a row where each word is a column creating a NXM matrix.



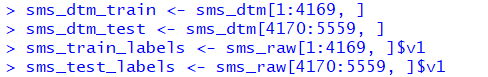
Let’s compare the two matrices.



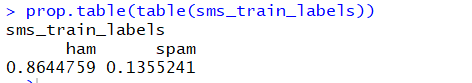


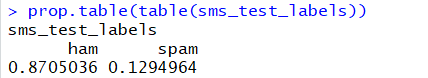
## Data Processing for Modeling:

Next, I will split the data into training and testing data as below:

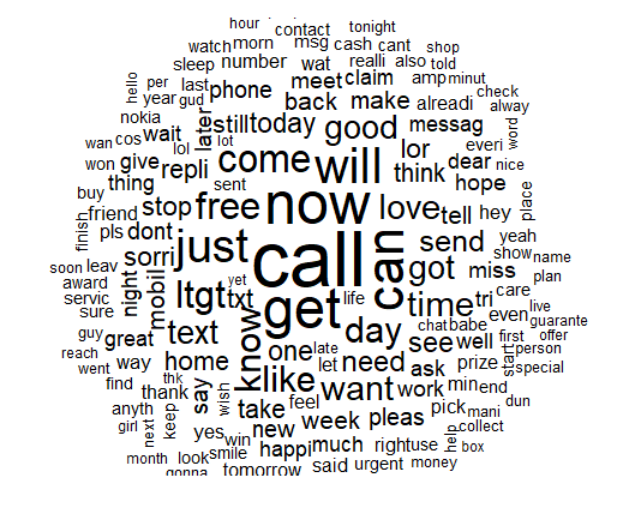


A proportion table will show if the training and testing data are both representative of the whole dataset.

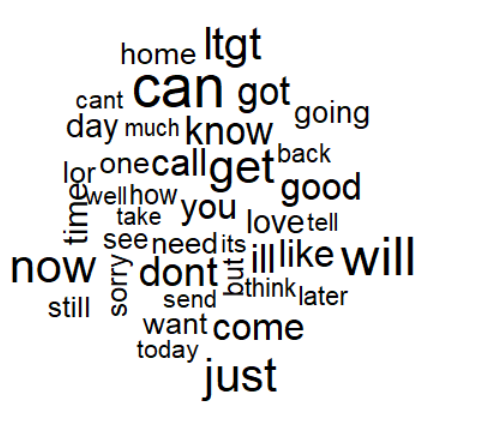




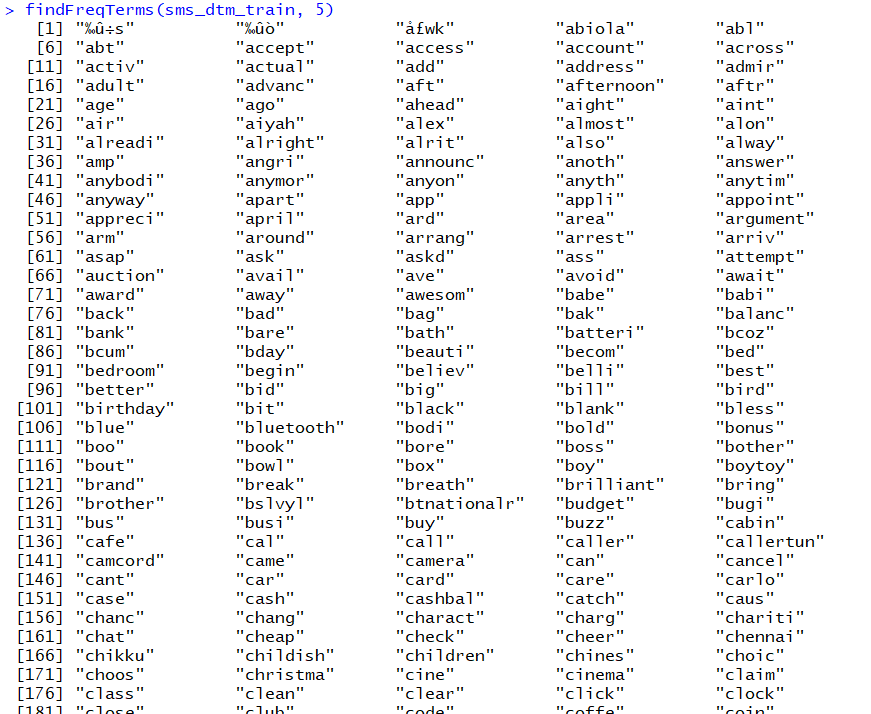
I will build a word cloud to visually look at our corpus to see the frequency of words within the corpus. Words that appear more frequently will be larger in font size and less frequent will be smaller in size. This can be done using the wordcloud package in R.



Next, I will use the subset() function to look at the individual wordcloud of the two categories of email as below:

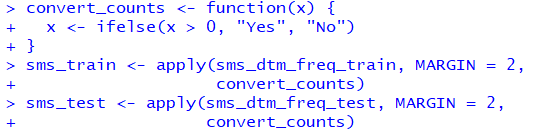


I will reduce the number of features in the training and testing data by using findFreqTerms() function.





I will write a function that will convert the sparse document matrices from numeric to categorical yes/no matrices that the algorithm can process.



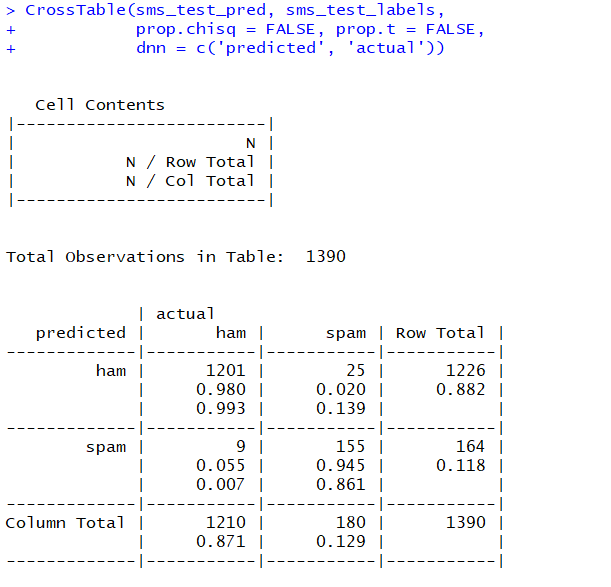
## Data Modeling:

Now, I will apply the naïve bayes classifier to the training set and make predictions using the testing set.



## Model Performance:

I will now use crosstable function from gmodels package in R to evaluate the predictions.



In the above model, we can see that the accuracy percentage achieved was 97.55%

## Conclusion:

The accuracy achieved for classifying the two categories of email using Naïve Bayes Algorithm was 97.5% which is very high accuracy percentage, so we can say that Naïve Bayes works fairly well in classifying text data.