Eva Diana Maria Conevski – 200083191

COS498 – Machine Learning with R

Prof. Dimitar Christozov

# SMS Spam Detection with Naive Bayes Classification

The objective of the project is to develop a predictive model that can be put to use in the process of recognizing spam SMS messages. Since phishing has emerged as a more significant security risk in recent years, I find this to be an engaging topic to discuss.

The dataset was obtained from Kaggle and can be found: <https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>. This data consisted of 5572 records, where 87% of them were legitimate text messages and the other 13% were unsolicited messages.

Graphical user interface, text

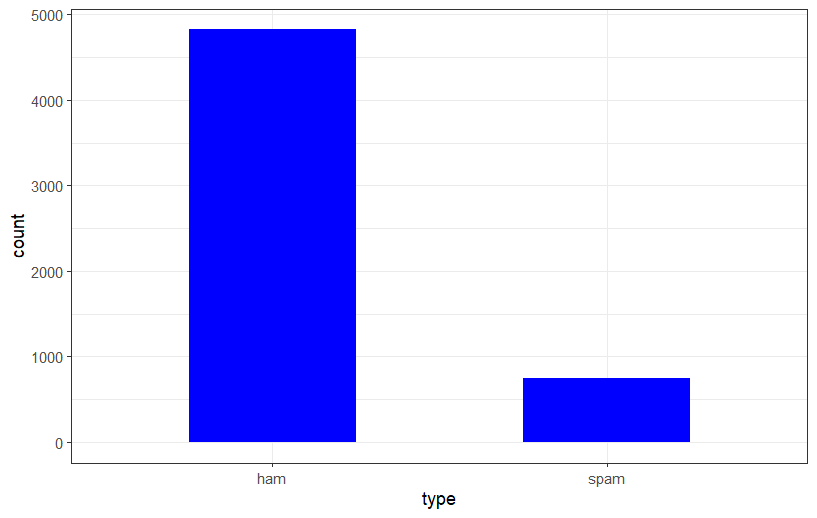
Description automatically generated

in the below paragraphs I will present the results that I have gotten from my model and explain the steps of the gotten results. The code for the project will be put at the end of the file.

## Loading and pre-processing the data

The dataset that I will be working with is a collection of SMS messages that are classified as spam or ham. The messages that are classified as ham are legitimate messages. The data had 5 columns, where only the first two were useful for my project. I have read the data into a data frame, then I renamed the first column from v1 to type since the column indicates the type of the message as being a spam or ham and the v2 column to text. Then I have also converted the column into a factor vector, so I can use it as a classifier label in the feature.

After I have made a bar chart of the data to have a better visualization of the count of spam and ham messages.



## Creating a document-term matrix

A document-term matrix is a mathematical matrix that describes the frequency with which particular terms appear in a particular set of documents. The rows of a document-term matrix correspond to the documents in the collection, and the columns of the matrix correspond to the terms.

In order to achieve this, I have created a source object used in VSource. The reason why I have done this is because the data can’t be dealt with directly in the data frame structure and needed to be converted as a volatile corpus meaning that it will be stored in memory and would be destroyed when the R object containing is destroyed.

The R text processing package called Corpus offers complete support for international text (Unicode). It has tools for normalizing and tokenizing text, searching for term occurrences, computing term occurrence frequencies, and reading data from newline-delimited JSON files (including n-grams).

Text

Description automatically generated with medium confidence

Each SMS in the volatile corpus or text document was in raw data format. Therefore, I needed to process it before applying it to the Naive Bayes classification algorithm. The data cleaning process in this case was:

* turn all the words into lower case
* remove numbers since they can’t be identified as neither ham nor spam
* removing a list of stop words such as: “the”, “for”, “a”, “an” … since they can’t be identified as neither spam nor ham
* removing punctuation
* removing various words with the same stem

After performing these steps, the corpus was transformed into a document-term matrix, where each column represents a unique word as a feature for an SMS.

## Splitting the data into training and testing set

Next the document-term matrix was split into a training set with 70% of the data and 30% as a testing set. After I made a pair of vectors with labels for each row in the training and testing matrices. To assure that I have split the data correctly and insure the performance of the algorithm I have used prop.table() to convert the number of spam and ham labels in both of the label vectors into fractional values.

Text

Description automatically generated

From the figure above we can see that the both the testing and training datasets contain around 13% spam messages.

Next, I have decided to make a create a word cloud in order to visualize better the most frequent words in the spam and ham dataset. For both of the datasets I have set to filter out word that appear at least 50 times which is 0.8% of the total unique words (6118). In the word cloud we will be able to see that the words which are most frequent will appear larger and, in the centre, and the words that appear less will be smaller and around the edges.

The most frequent words in the spam dataset were:

Text

Description automatically generated



As we can see from the spam word cloud, the most frequent words were: call, now, txt, free, prize, mobile, urgent, won, guaranteed and so on.

The most frequent words in the ham dataset were:

Text

Description automatically generated

As we can see from the ham word cloud, the most frequent words were: can, get, will, know, good, love, come, call etc.

Something that I have noticed from the data is the word call appears in both spam and ham dataset as frequent, but the other words are quite distinct meaning that the algorithm should be able to give a good estimate of a ham or spam message.

## Reducing the number of features and converting the input

At the moment the matrix includes 6118 unique features which makes the matrix quite large. This is why I have decided to reduce this number by using only words that appear in 30 or more messages. In order to do this I have first created a vector of words where the frequency is larger than 30 and then I have filtered out the matrix to include only those words. From this I was left with a vector of 266 words. Using this information, I have selected and worked with only these frequent words meaning that the number of the training and testing dataset has shrined.

Both of the test and train document-term matrices contain numerical values representing the number of occurrences of each word in each SMS. In order to use the Naive Bayes algorithm, I needed to convert my data into a categorical input. To achieve this, I have created a function that takes in each numerical value and converts the value to “Yes” or “No” based on the situation. For example, if the value is 0 the cell will be converted to “No”, and if the value is greater than 0 it will be converted to “Yes”. Finally, I have used the apply function to set the values in the matrices.

## Training the model and evaluating the performance

In the last step I have applied the test dataset to the Naïve Bayes algorithm and applied the function predict in order to get the results of the training data. The results showed that there were 11 false positive cases meaning that legitimate messages were classified as spam and 22 false negative cases meaning that spam messages were classified as legitimate messages. I have put these results in a table to give a better visual representation:

Table

Description automatically generated

Chart, treemap chart

Description automatically generated



From this I have calculated the accuracy of my model being 98%.

Formula for calculations:

(number of cases from test dataset – errors) / number of cases from test dataset \* 100

or (1672-33)/1672 \* 100 = 98.03

## Conclusion

The project's goals were accomplished, and while the resulting model was successful, there is still room for improvement. My experience with this project has taught me that the messages that constitute spam are typically made up of a predetermined set of words. In addition, the presence of more than a few of these words in a message increases the likelihood that the message is spam.

## Code

options(warn = -1)

library(tidyverse)

install.packages('tm')

install.packages('Rtools')

library(Rtools)

install.packages('SnowballC')

library(SnowballC)

library(tm)

install.packages('wordcloud')

library(wordcloud)

library(e1071)

install.packages('gmodels')

library(gmodels)

library(ggplot2)

sms\_raw <- read.csv("../data.csv")

col\_types = cols(X = col\_skip(), X.1 = col\_skip(), X.2 = col\_skip(), v1 = col\_factor(levels = c("spam", "ham")))

sms\_raw <- rename(sms\_raw, type = v1, text = v2)

sms\_raw <- subset(sms\_raw, select = -c(X, X.1, X.2))

table(sms\_raw$type)

ggplot(aes(x=type),data=sms\_raw) +

geom\_bar(fill="blue",width=0.5)

sms\_raw[, "text"] <- gsub("[^[:alnum:]]", " ", sms\_raw$text)

col\_types = cols(type = col\_factor(levels = c("spam", "ham")))

sms\_corpus <- VCorpus(VectorSource(sms\_raw$text))

inspect(sms\_corpus[1:2])

as.character(sms\_corpus[[1]])

lapply(sms\_corpus[1:2], as.character)

replacePunctuation <- function(x) { gsub("[[:punct:]]+", " ", x) }

sms\_dtm <- DocumentTermMatrix(sms\_corpus, control = list(tolower = TRUE, removeNumbers = TRUE, removePunctuation = TRUE, stemming = TRUE))

dim(sms\_dtm)

sms\_dtm\_train <- sms\_dtm[1:3900, ]

sms\_dtm\_test <- sms\_dtm[3901:5572, ]

sms\_train\_labels <- sms\_raw[1:3900, ]$type

sms\_test\_labels <- sms\_raw[3901:5572, ]$type

prop.table(table(sms\_train\_labels))

prop.table(table(sms\_test\_labels))

spam <- subset(sms\_raw, type == "spam")

ham <- subset(sms\_raw, type == "ham")

wordcloud(spam$text, min.freq = 50, max.words = 60, colors = brewer.pal(6, 'Dark2'), random.order = FALSE)

wordcloud(ham$text, min.freq = 50, max.words = 60, colors = brewer.pal(6, 'Dark2'), random.order = FALSE)

sms\_freq\_words <- findFreqTerms(sms\_dtm\_train, 30)

sms\_dtm\_freq\_train <- sms\_dtm\_train[ , sms\_freq\_words]

sms\_dtm\_freq\_test <- sms\_dtm\_test[ , sms\_freq\_words]

str(sms\_freq\_words)

convert\_counts <- function(x) {

x <- ifelse(x>0, "Yes", "No")

}

sms\_train <- apply(sms\_dtm\_freq\_train, MARGIN = 2, convert\_counts)

sms\_test <- apply(sms\_dtm\_freq\_test, MARGIN = 2, convert\_counts)

sms\_classifier <- naiveBayes(sms\_train, sms\_train\_labels)

sms\_test\_pred <- predict(sms\_classifier, sms\_test)

CrossTable(sms\_test\_pred, sms\_test\_labels,

prop.chisq = FALSE,

prop.c = FALSE,

prop.r = FALSE,

dnn = c('predicted', 'actual'))

conf\_matrix <- as.data.frame(table(sms\_test\_pred, sms\_test\_labels))

ggplot(conf\_matrix, aes(sms\_test\_pred, sms\_test\_labels, fill = Freq)) + geom\_tile() + geom\_label(aes(label = Freq)) + scale\_fill\_gradient(low = "#fcc368", high = "#6877fc", trans = "log")