PROJECT REPORT

OF

MACHINE LEARNING

SMS SPAM FILTER

(USING R)

Made By: **Group-8**

Yogesh Sharma

Nikhil Dhiman

Archit Puri

Akash

Chetanya Nagpal

SMS SPAM FILTER USING ML

INTRODUCTION:

A ***SMS Spam Filter*** is a program that is used to detect unsolicited and unwanted sms and prevent those messages from getting to a user's inbox. Like other types of filtering programs, a spam filter looks for certain criteria on which it bases judgments.

The simplest and earliest versions (such as the one available with Microsoft's Hotmail) can be set to watch for particular words in the subject line of messages and to exclude these from the user's inbox. This method is not especially effective, too often omitting perfectly legitimate messages (these are called false positives) and letting actual spam through. More sophisticated programs, such as Bayesian filters or other heuristic filters, attempt to identify spam through suspicious word patterns or word frequency.

OBJECTIVE:

Spam is annoying, no doubt, but it can also be dangerous. Malware and phishing are hugely profitable for scammers and can be costly for mailbox providers' customers, as well as the mailbox providers who face intense market competition. Practically speaking, spam filters drastically reduce the load on server resources, considering that 70 percent of all mail sent globally is spam.

If you have had similar experiences of being bombarded with text-messages for marketing purposes, then this post may be of interest. I will use R and the TM (text mining) package to build a text-message Spam Filter Machine Learning model by means of a Naïve Bayes algorithm, to predict which messages would be classified as either spam or genuine text-messages.

R CODE:

```
#Installing required packages

install.packages("gmodels")

install.packages("e1071")

install.packages("wordcloud")

install.packages("tm")

install.packages("SnowballC")


# Importing the data

sms_raw <- read.csv("sms_spam.csv", stringsAsFactors = FALSE)

str(sms_raw)

sms_raw$type <- factor(sms_raw$type)

str(sms_raw$type)

table(sms_raw$type)


# Using "tm" library (TEXT MINING)

#Text Data Preparation

library(tm)

sms_corpus <- VCorpus(VectorSource(sms_raw$text))

print(sms_corpus)

inspect(sms_corpus[1:2])

as.character(sms_corpus[[1]])

lapply(sms_corpus[1:2], as.character)

sms_corpus_clean <- tm_map(sms_corpus, content_transformer(tolower))

as.character(sms_corpus[[1]])

as.character(sms_corpus_clean[[1]])

sms_corpus_clean <- tm_map(sms_corpus_clean, removeNumbers) # To remove numbers
```

```
sms_corpus_clean <- tm_map(sms_corpus_clean, removeWords, stopwords()) # To remove stop words

sms_corpus_clean <- tm_map(sms_corpus_clean, removePunctuation) # To remove punctuation

removePunctuation("hello.world")

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

replacePunctuation("hello.world")


# Using "SnowballC" library for text data preparation

library(SnowballC)

wordStem(c("learn", "learned", "learning", "learns"))

sms_corpus_clean <- tm_map(sms_corpus_clean, stemDocument)

sms_corpus_clean <- tm_map(sms_corpus_clean, stripWhitespace) # To eliminate unneeded whitespace

lapply(sms_corpus[1:3], as.character)

lapply(sms_corpus_clean[1:3], as.character)

sms_dtm <- DocumentTermMatrix(sms_corpus_clean)

sms_dtm2 <- DocumentTermMatrix(sms_corpus, control = list(

  tolower = TRUE,

  removeNumbers = TRUE,

  stopwords = TRUE,

  removePunctuation = TRUE,

  stemming = TRUE

))

sms_dtm3 <- DocumentTermMatrix(sms_corpus, control = list(

  tolower = TRUE,

  removeNumbers = TRUE,

  stopwords = function(x) { removeWords(x, stopwords()) },

  removePunctuation = TRUE,
```

```
  stemming = TRUE
))
sms_dtm
sms_dtm2
sms_dtm3
sms_dtm_train <- sms_dtm[1:4169, ]
sms_dtm_test <- sms_dtm[4170:5559, ]
sms_train_labels <- sms_raw[1:4169, ]$type
sms_test_labels <- sms_raw[4170:5559, ]$type
prop.table(table(sms_train_labels))
prop.table(table(sms_test_labels))

# Using "wordcloud" library WordCloud is a package for visualizing text data.
# The larger Bold words represented occur more frequently whereas the smaller less Bold words do not appear as often.

library(wordcloud)
wordcloud(sms_corpus_clean, min.freq = 50, random.order = FALSE)

# subset the training data into spam and ham groups
spam <- subset(sms_raw, type == "spam")
ham <- subset(sms_raw, type == "ham")
wordcloud(spam$text, max.words = 40, scale = c(3, 0.5))
wordcloud(ham$text, max.words = 40, scale = c(3, 0.5))

# Frequent word indicators
sms_dtm_freq_train <- removeSparseTerms(sms_dtm_train, 0.999)
sms_dtm_freq_train
```

```
findFreqTerms(sms_dtm_train, 5)

sms_freq_words <- findFreqTerms(sms_dtm_train, 5)

str(sms_freq_words)

sms_dtm_freq_train <- sms_dtm_train[ , sms_freq_words]

sms_dtm_freq_test <- sms_dtm_test[ , 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)


# Training the model

library(e1071)

sms_classifier <- naiveBayes(sms_train, sms_train_labels)



sms_test_pred <- predict(sms_classifier, sms_test)


# Evaluating and improving the performance of the model

library(gmodels)

CrossTable(sms_test_pred, sms_test_labels,

           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,

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

sms_classifier2 <- naiveBayes(sms_train, sms_train_labels, laplace = 1)

sms_test_pred2 <- predict(sms_classifier2, sms_test)

CrossTable(sms_test_pred2, sms_test_labels, prop.chisq = FALSE,

           prop.t = FALSE, prop.r = FALSE,dnn = c('predicted', 'actual'))
```

Procedure Followed:

1. Importing data into R environment:

```
> # Importing the data
> sms_raw <- read.csv("sms_spam.csv", stringsAsFactors = FALSE)
> str(sms_raw)
'data.frame':	5574 obs. of  2 variables:
 $ type: chr  "ham" "ham" "spam" "ham" ...
 $ text: chr  "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat..." "O
k lar... Joking wif u oni..." "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive ent
ry question("| __truncated__ "U dun say so early hor... U c already then say..." ...
> sms_raw$type <- factor(sms_raw$type)
> str(sms_raw$type)
 Factor w/ 2 levels "ham","spam": 1 1 2 1 1 2 1 1 2 2 ...
> table(sms_raw$type)

 ham spam
4827  747
```

The data contains two variables "type" variable contains either "ham" or "spam" referring to either a genuine text-message or a spam message respectively. The "text" variable contains the actual wording of the message.

2. Text data preparation:

```
> # Using "tm" library (TEXT MINING)
> #Text Data Preparation
> library(tm)
Loading required package: NLP
Warning message:
package 'tm' was built under R version 3.5.1
> sms_corpus <- VCorpus(VectorSource(sms_raw$text))
> print(sms_corpus)
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 5574
> inspect(sms_corpus[1:2])
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 111

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 29
```

Before analytical efforts can be made on this data we will need to standardize it into a format that can be understood by the Machine. As you would imagine, a text-message contains full sentences with characters, spacing, numbers etc. We need to create a dataset in the traditional row-column space (structured data) from this unstructured data. To do this we will need to remove Capital letters, punctuation like commas or full stops, delimiters like spaces or tabs, to result in a single row per text message with each word in a unique column.

3. Visualizing text data:

   WordCloud is a fun package for visualizing frequency, or prevalence, of words in text data. The larger Bold words occur more frequently, where the smaller less Bold words do not appear as often.

   Output:

From the figure the word “Call” stands out and can be thought of as appearing frequently. The word “box”, for example, is much smaller and therefore does not appear often in the data.

4. Frequent word indicators:

```
> # Frequent word indicators
> sms_dtm_freq_train <- removeSparseTerms(sms_dtm_train, 0.999)
> sms_dtm_freq_train
<<DocumentTermMatrix (documents: 4169, terms: 1123)>>
Non-/sparse entries: 25065/4656722
Sparsity           : 99%
Maximal term length: 13
Weighting          : term frequency (tf)
> findFreqTerms(sms_dtm_train, 5)
   [1] "â£wk"        "â€¦"           "â€“"          "abiola"      "abl"         "abt"         "accept"
   [8] "access"      "account"       "across"      "activ"       "actual"      "add"         "address"
  [15] "admir"       "adult"         "advanc"      "aft"         "afternoon"   "aftr"        "age"
  [22] "ago"         "ahead"         "aight"       "aint"        "air"         "aiyah"       "alex"
  [29] "almost"      "alon"          "alreadi"     "alright"     "alrit"       "also"        "alway"
  [36] "amp"         "angri"         "announc"     "anoth"       "answer"      "anybodi"     "anymor"
  [43] "anyon"       "anyth"         "anytim"      "anyway"      "apart"       "app"         "appli"
  [50] "appoint"     "appreci"       "april"       "ard"         "area"        "argument"    "arm"
  [57] "around"      "arrang"        "arrest"      "arriv"       "asap"        "ask"         "askd"
  [64] "asleep"      "ass"           "attempt"     "auction"     "avail"       "ave"         "avoid"
  [71] "await"       "award"         "away"        "awesom"      "babe"        "babi"        "back"
  [78] "bad"         "bag"           "bak"         "balanc"      "bank"        "bare"        "bath"
  [85] "batteri"     "bcoz"          "bcum"        "bday"        "beauti"      "becom"       "bed"
  [92] "bedroom"     "begin"         "believ"      "belli"       "best"        "better"      "bid"
  [99] "big"         "bill"          "bird"        "birthday"    "bit"         "black"       "blank"
 [106] "bless"       "blue"          "bluetooth"   "bodi"        "bold"        "bonus"       "boo"
 [113] "book"        "bore"          "boss"        "bother"      "bout"        "bowl"        "box"
 [120] "boy"         "boytoy"        "brand"       "break"       "breath"      "brilliant"   "bring"
 [127] "brother"     "bslvyl"        "btnationalr" "budget"      "bugi"        "bus"         "busi"
 [134] "buy"         "buzz"          "cabin"       "cafe"        "cal"         "call"        "caller"
 [141] "callertun"   "camcord"       "came"        "camera"      "can"         "cancel"      "cant"
 [148] "car"         "card"          "care"        "carlo"       "case"        "cash"        "cashbal"
 [155] "catch"       "caus"          "chanc"       "chang"       "charact"     "charg"       "chariti"
 [162] "chat"        "cheap"         "check"       "cheer"       "chennai"     "chikku"      "childish"
 [169] "children"    "chines"        "choic"       "choos"       "christma"    "cine"        "cinema"
 [176] "claim"       "class"         "clean"       "clear"       "click"       "clock"       "close"
 [183] "club"        "code"          "coffe"       "coin"        "cold"        "colleagu"    "collect"
 [190] "colleg"      "colour"        "come"        "comin"       "comp"        "compani"     "competit"
 [197] "complet"     "complimentari" "comput"      "concentr"    "condit"      "confid"      "confirm"
 [204] "congrat"     "congratul"     "connect"     "contact"     "content"     "convey"      "cook"
 [211] "cool"        "copi"          "correct"     "cos"         "cost"        "countri"     "coupl"
 [218] "cours"       "cover"         "coz"         "crave"       "crazi"       "credit"      "cri"
 [225] "croydon"     "cuddl"         "cum"         "cup"         "current"     "custcar"     "custom"
 [232] "cut"         "cute"          "cuz"         "dad"         "daddi"       "damn"        "darl"
 [239] "darlin"      "darren"        "dat"         "date"        "day"         "dead"        "deal"
 [246] "dear"        "decid"         "deep"        "definit"     "del"         "delet"       "deliv"
 [253] "deliveri"    "den"           "depend"      "detail"      "dev"         "didnt"       "die"
```

Words that appeared less than a certain number of times within the full data should be removed in order to exclude noise from the data. Arbitrarily, this example excluded words that appeared less than 5 times.

5. Training the model:

```
> library(e1071)
Warning message:
package ‘e1071’ was built under R version 3.5.1
> sms_classifier <- naiveBayes(sms_train, sms_train_labels)
>
>
> sms_test_pred <- predict(sms_classifier, sms_test)
>
> # Evaluating and improving the performance of the model
> library(gmodels)
Warning message:
package ‘gmodels’ was built under R version 3.5.1
> CrossTable(sms_test_pred, sms_test_labels,
+            prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
+            dnn = c('predicted', 'actual'))


   Cell Contents
|-------------------------|
|                       N |
|           N / Col Total |
|-------------------------|
```

The raw text data has been transformed into a format that can be understood by the Machine. The Naïve Bayes algorithm can now be applied to the data to predict which messages represent Spam or genuine text-messages.

6. Evaluating and improving the performance of the model:

```
Total Observations in Table:  1390


             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |      1200 |        20 |      1220 |
             |     0.993 |     0.110 |           |
-------------|-----------|-----------|-----------|
        spam |         9 |       161 |       170 |
             |     0.007 |     0.890 |           |
-------------|-----------|-----------|-----------|
Column Total |      1209 |       181 |      1390 |
             |     0.870 |     0.130 |           |
-------------|-----------|-----------|-----------|


> sms_classifier2 <- naiveBayes(sms_train, sms_train_labels, laplace = 1)
> sms_test_pred2 <- predict(sms_classifier2, sms_test)
> CrossTable(sms_test_pred2, sms_test_labels, prop.chisq = FALSE,
+            prop.t = FALSE, prop.r = FALSE,dnn = c('predicted', 'actual'))


   Cell Contents
|-------------------------|
|                       N |
|           N / Col Total |
|-------------------------|


Total Observations in Table:  1390


             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |      1202 |        28 |      1230 |
             |     0.994 |     0.155 |           |
-------------|-----------|-----------|-----------|
        spam |         7 |       153 |       160 |
             |     0.006 |     0.845 |           |
-------------|-----------|-----------|-----------|
Column Total |      1209 |       181 |      1390 |
             |     0.870 |     0.130 |           |
-------------|-----------|-----------|-----------|
```

Accuracy of model = [{(1202+153)-(28+7)}/1390]*100 = 94.96%

References:

www.google.com

www.rdocumentation.org

www.techopedia.com

https://cran.r-project.org