The University of Akron

College of Business, Department of Management

Advanced Data Analytics Topics (ISM:663)

Project 2

Spam message detection using Naïve Bayes Algorithm

By:

Parwan Sharma

Student id – 4808380

MSM Business Analytics

Abstract

The project is based on the use of the Naïve Bayes Algorithm, which is largely used for credit scoring, medical data classification, weather forecasting etc. This Project will cover the basic principles of probability, data structures which are analyzed with the help of R using different specialized methods, and how to use the Naïve Bayesian algorithm to create a data filter that can filter junk messages. The data set used for this analysis is sms\_spam.csv. The data is first standardized, cleaned(removing numbers, punctuations, etc), stemmed and processed. After the data is prepared, it is visualized with “word clouds”. The model will be trained and evaluated to get the best results.

Introduction

The Naïve Bayesian Algorithm is based on the idea that the estimated outcome or the potential likelihood of an event (known as probability) should be based upon the evidence available through a series of multiple trials for that particular event.

This probability is calculated by creating a ratio of the favorable outcomes and the total number of outcomes. For Example, when you throw a die, the probability of getting any favorable outcome, for example, 4 as the number would be, 1/6 = 0.166.

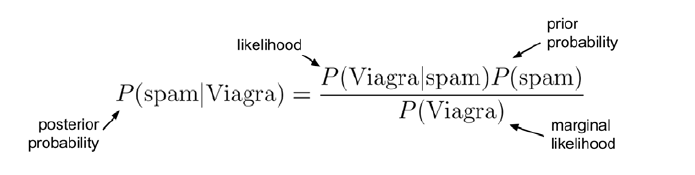
The above-mentioned example explains a mutually exclusive and exhaustive case of probability, i.e. an event where both events cannot happen simultaneously. But in case of situations where the event is not mutually exclusive, we can make a prediction of an event of interest. Using the Bayes algorithm for an event A, given that the event B has already occurred is denoted by P(A|B) (which is also known as the conditional probability), we can get an estimate of P(A|B) based on P(A ∩ B)

Text

Description automatically generated with medium confidence

To understand this equation, we can use the example of spam words such as *Viagra.* We can use the probability of this word found in the spam messages and check how the probability of the existence of the word Viagra, P(Viagra) in a message affects the probability of the message being spam P(Viagra|Spam) or not.

By Applying the Bayes theorem, we can calculate the posterior probability, we can calculate the probability message to be spam.



The Naïve Bayes algorithm describes a simple method to apply the Bayes theorem to classify the above mentioned problem. The algorithm deals well with data that are noisy and have missing data points, hence one of the best approaches to filter spam messages or events such as weather forecasting. It is simply fast and very effective and requires few training examples to obtain an estimated probability for a prediction.

Prompt

In this case, we analyze a dataset called sms\_spam dataset and make a report by applying the Naïve Bayesian algorithm. We process the plain text file using the tm\_mpa() function in the tm package in R. We will perform stemming, changing case, removing stop words, numbers, whitespaces and split the text messages into a document matrix followed by the data visualization using a word cloud. The model is later trained and evaluated in order to be improved.

Problem Description

The SMS service revolutionized the mode of communication. It was precise and instantaneous making it soon a preferred choice of communication for a lot of people. With the growing popularity, soon people started using the platform as a source of advertisement. This soon became a cause of people receiving an influx of unwanted or irrelevant messages known as spam messages. Since a load of messages was growing exponentially, it was required to filter out spam messages for the overall efficiency of the system.

There was also the factor of the health of population as on many occasions the messages can be inappropriate for certain groups of people, for example children. This made spam messages a cause of trauma and nuisance among people. There was also the fact that people were charged for incoming messages. Hence no one wanted to pay for a message that is irrelevant or inappropriate. This demands use of a tool that can effectively filter out spam messages.

The shorthand lingo, lack of words(since the number of words entered in a message was limited to 160 characters) and a number of error in a text message made Naïve Bayes Algorithm an effective way to analyze the text as the algorithm is simple, fast and effective and can deal with noisy and missing data and can be trained with few examples.

Method

This project will be using the dataset sms\_csv. The primary source of literature used Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8). Listed below are the steps taken in the report:

1. Collecting data
2. Exploring and preparing the data
3. Cleaning and Standardizing text data
4. Splitting text Documents into words
5. Creating training and test data set
6. Visualizing text data- word cloud
7. Creating indicator features for frequent words.
8. Training a model on the data
9. Evaluating model performance
10. Improving model performance

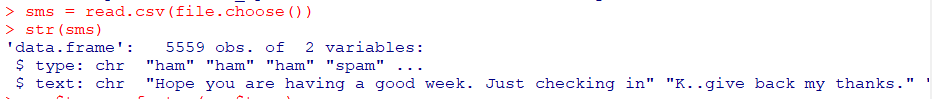
Steps taken:

Step 1 Collecting data.

We use the open-source dataset called “sms\_spam.csv” and use it in R as it contains a collection of SMS messages labeled as “spam” or “ham”. We use this dataset for building and evaluating a model based on Naïve Bayes for spam detection.

Step 2 Exploring and preparing the data.

The dataset is big in size as it contains 5559 SMS. Data has two characteristics, that is type and text. Type indicates if the message is “spam” or “ham”, and the text contains the full text of SMS.



Since both the elements are character types and not categorical, we will have to convert them into factors to use the algorithm. We use the table() function and see a total of 4812 ham and 747 spam messages.

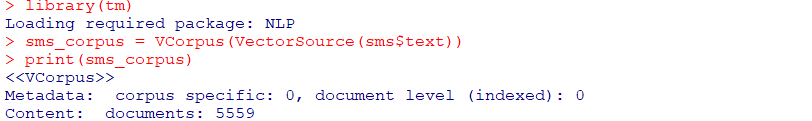
Chart

Description automatically generated with medium confidence

Step 3. Cleaning and Standardizing text data

Just like the usual SMS format, this data is in the raw form, and it contains punctuations, random spaces and errors, noise and, numbers. We use the tm(Text Mining) package in R to manipulate, manage and mine in order to clean visualize and transform it.

First, we create corpus to with the help of VCorpus() function. In this case we create a source object from existing vector sms$text which is then provided to VCorpus. It is saved as sms\_corpus which now contains all the training data.



The inspect() function is used to summarize specific messages from sms\_corpus. In the below-mentioned case it is the summary of the first sms\_corpus.

Graphical user interface, text, application

Description automatically generated with medium confidence

The as.character is used, to see the exact text for the first message. lapply is used for all the elements of the dataset.

Graphical user interface, application

Description automatically generated

Using tm\_map() function for cleaning the data. The cleaned ruslts are saved in sms\_corpus\_clean. Tolower()function is used to convert all upper case to lower case. content\_transformis used for transformation. Text is checked again.



Now that all the text is in lower case, we can move forward to remove numbers using the removeNumber() function. Stop words such as ‘ as, why, etc’ are removed by removeWords command. Similarly we use removePunctuation() function to remove punctuations

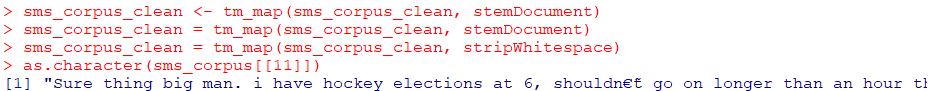
A picture containing text

Description automatically generated

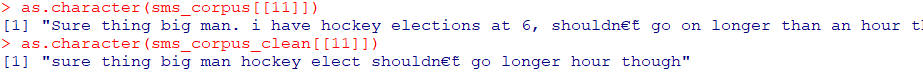
We use the process called stemming,which is a part of SnowballC package. We use it to convert text to its root form making the data uniform.



The final step of cleaning is to remove whitespace using stripWhitespace().



Below is the difference between the 11th message of the cleaned and uncleaned sms\_corpus.



Step 4 Splitting text document into text

We use document term matrix(DTM), a matrix in which the columns contain words and row contains SMS messages.

A picture containing text

Description automatically generated

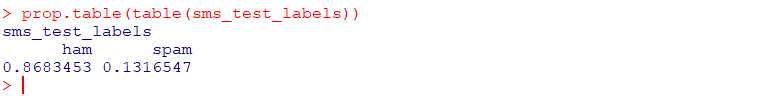
The matrix is named sms\_dtm2 which includes all the cleaning of the data. A number of cells in DTM represent 0 as certain words do not appear in many messages. Order worked well and show a total of 43221 terms in matrix.

Graphical user interface, text

Description automatically generated

Step 5 Creating training and test data set.

Now that the data is prepared, the data is split into training and test set. The ratio for the split is 75% is training set and 25% is training set. This lead to a random split of 4169 text for the training set and 1390 for the testing set. The sets are named accordingly as mentioned below. Both the data set has the same amount of spam words, which makes the ideal for evaluation.



Step 6 Visualizing text data- word cloud

The word cloud is used as a visual representation of the frequency of word that are present in the data set. The Representation is in such a way that the bigger the number is in the visualization, the more frequently it appears in the dataset. It is also signified by the positioning of the word. The closer the word is to the center of the cloud, the bigger it would appear, signifying higher frequency.

The minimum frequency is set to 50, i.e. the word should have a minimum frequency of 50 in order to appear in the cloud.



Text

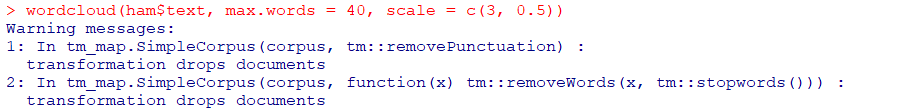
Description automatically generated

Tow word cloud for the sms is also created by splitting the data based on the existence of the word “spam” and “ham” and two subsets are created. 40 of the most common words are selected in both the subsets with a minimum frequency of 50 words.



Text

Description automatically generated



Text

Description automatically generated

Step 7 Creating indicator features for frequent words.

Now the convert\_count function is used to check the words and mark them as “Yes” or “No”. This is done as the matrix we created was using numeric values, whereas the Naïve Bayes algorithm uses categorical elements.

In this function, if the value of x is greater than zero, it will replace it with “Yes” and if not then with a “No”



We use apply function to apply this function to all the columns.



Table

Description automatically generated

Step 8 Training a model on the data.

In this step, we first install the “e1071” package. Now that the messages are in a proper format to be used in a Naïve Bayes algorithm. We can use it to analyze the presence or absence of a word to evaluate the probability if a message is a spam or not.

Step 9 Evaluating model performance.

The table below indicates that:

* 30 out of 183 messages were improperly marked as ham(False Positive).
* 6 out of 1207 are misidentified as spam(False Negative).
* 36 messages had wrong classification(FP+FN)
* 153 cases are True negative.
* Model identified 11.4% of spam and 88.6% of ham messages whereas the actual number of spam messages are 13.2% and actual number of ham messages are 86.8%.

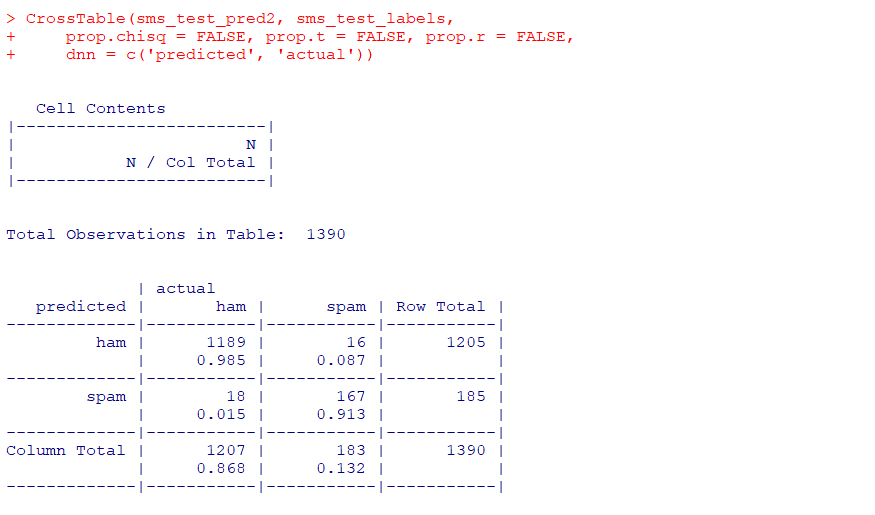
Table

Description automatically generated

Step 10 Improving model performance.

In order to improve the data we will use the Laplace estimator to calculate the new values again. The estimator will add 1 to each class and divide the sum by total number in class plus number of unique features. Therefor by adding 1 to each class it will change the 0 probability, making the analysis more effective and reliable.

This time we use Laplace as one and build sms\_test\_pred2 to create a new table.



Outcomes when compared to the old table:

* Error in the classification is reduced by 2(36-34[18FP+16FN]).
* False Positive increased by 12(18-6).
* False Negative reduced by 14(30-16).

Finding

The Naïve Bayes algorithm is one of the most common machine learning algorithms which is based on probability. It is useful when in a probability we can observe evidence that a probability of a prior event can affects the overall probability of an outcome. For example, in this case we use the probability of the word *“Viagra”* and related its probability using the Naïve Bayes to classify if the message containing this word is actually a spam message or not. This makes the predictive analyses of this event effective as we were able to predict if a message is a spam or not based on the occurrence of the word “Viagra”. By evaluating the model, we can study the effectiveness of the prediction which can be improved.

Conclusion

We construct a table based on the Naïve Bayes algorithm which gives us the likelihood of the example belonging to various class. By using the tools such as map function, dtm matrices, altering and changing data and using word cloud visualization we were able to analyze a large dataset efficiently and effectively. We use these tools to create a model using the Naïve bayes algorithm which is highly accurate and can classify if a message is a spam or not. This report proves that the Naïve Bayes algorithm is very valuable tool that can be a highly efficient probability-oriented predictive model to analyse large data set such as text messages. It demonstrates how it can deal with big dataset, with noise and gaps accurately making it a highly used solution for such problems.

Coding

* Exploring and Preparing

> sms = read.csv(file.choose())

> str(sms)

> sms$type = factor(sms$type)

> str(sms$type)

> table(sms$type)

* Data Preparation

> install.packages("tm")

> library(tm)

> sms\_corpus = VCorpus(VectorSource(sms$text))

> print(sms\_corpus)

> inspect(sms\_corpus[1])

> 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\_clean[[1]])

>sms\_corpus\_clean= tm\_map(sms\_corpus\_clean, removeNumbers)

>sms\_corpus\_clean= tm\_map(sms\_corpus\_clean,removeWords, stopwords())

>sms\_corpus\_clean=tm\_map(sms\_corpus\_clean, removePunctuation)

Install.packages(“SnowballC”)

Llibrary(SnowballC)

sms\_corpus\_clean = tm\_map(sms\_corpus\_clean, stemDocument)

sms\_corpus\_clean = tm\_map(sms\_corpus\_clean, stripWhitespace)

* Splitting into words

sms\_dtm = DocumentTermMatrix(sms\_corpus\_clean)

* Creating Training and Test Data sets

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[4170:5559, ]$type

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

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

sms\_train\_labels

* Visualizing Data

> install.packages("wordcloud")

> library(wordcloud)

> wordcloud(sms\_corpus\_clean, min.freq = 50, random.order = FALSE)

> spam = subset(sms, type == "spam")

> ham = subset(sms, type == "ham")

> wordcloud(spam$text, max.words = 40, scale = c(3, 0.5))

> wordcloud(ham$text, max.words = 40, scale = c(3, 0.5))

* Creating and indicator Features

> 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 a model data

> install.packages("e1071")

> library(e1071)

> sms\_classifier = naiveBayes(sms\_train, sms\_train\_labels)

* Evaluating Model Performance

> sms\_test\_pred = predict(sms\_classifier, sms\_test)

> library(gmodels)

> CrossTable(sms\_test\_pred, sms\_test\_labels, prop.chisq = FALSE, prop.t = FALSE, dnn = c('predicted', 'actual'))

* Improving Model Performance

> 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'))

Reference

* Machine Learning with R, by Brett Lantz, 2nd Ed., Packet Publishing, 2015 (ISBN: 978-1-78439-390-8)