naive_bayes_project

Manuel Herrera Lara y Anahí Berumen Murillo

21/11/2020

Perform a basic data analysis describing the dataset, summary statistics, data distribution, etc.

The data domain

We are going to apply the naive bayes algorithm to a dataset that contains comments or tweets directed towards the president of the United States, Donald Trump. In order to classify tweets as positive or negative; and thus achieve to have in context the opinions of the people towards the president. The tweets published by President Donald Trump are related to the 2020 presidential election process in the United States, and which is currently taking place. In this election process the candidates for the presidency are Donald Trump and Joe Biden; winning as president-elect Joe Biden. And finally we must emphasize that the algorithm to use "naive bayes" is used to classify everything that is text and is based on probability.

How the data was recollected, limitations of the study, disadvantages, etc.

The data was collected from the social network Twitter and from these tweets we formed a dataset with 204 observations. The tweets are related to the United States election process, in which Joe Biden appeared as president-elect and Donald Trump alleges fraud.

Description of the variables of the dataset

type Indicates the classification of the tweet. (P = Positivo y N = Negativo)
tweet Contains the text of the comment

» (dataset reading)

```
knitr::opts_chunk$set(echo = TRUE)
# path of the dataset
setwd("/home/chino/Documentos/17_materias_IS/1_mineria_de_datos/10_semana_miniproyecto3/1_algoritmo_nai

# read the dataset
tweets_de_trump <- read.csv("tweets_donald_trump.csv", stringsAsFactors = FALSE)
```

Basic summary statics

- It shows the first 10 records of the dataset.

```
head(tweets_de_trump, 10)
```

```
##    type
## 1     N
## 2     N
## 3     N
## 4     P
## 5     P
## 6     P
## 7     P
## 8     P
## 9     P
## 10    N
##
## 1
## 2
## 3
## 4                        Jesus wanted everyone to be equal and United as people. Socialism is exactl
## 5
## 6
## 7                                                                  7.5 trillion added to national debt
## 8  It couldnt have anything do with the increase in knowledge and general information given to all v
## 9                                                      Biden has 80 million votes. Do you beli
## 10
```

- It shows the structure of the data and/or the data types of the attributes.

```
str(tweets_de_trump)
```

```
## 'data.frame':    204 obs. of  2 variables:
##  $ type : chr  "N" "N" "N" "P" ...
##  $ tweet: chr  "And strike and strike and strike and strike and strike! Aama lama rama lama bimbam!"
```

Describe the distribution of the data.

Exploring the Variables

```
table(tweets_de_trump$type)
```

Tweets

```
##
##   N   P
## 132  72
```

```
tweets_table <- table(tweets_de_trump$type)
tweets_pct <- prop.table(tweets_table) * 100
round(tweets_pct, digits = 1)
```

Percentage of tweets

```
##
##    N    P
## 64.7 35.3
```

Application of the naive bayes algorithm

step 2: Exploring and preparing the data

```
# dataset path
setwd("/home/chino/Documentos/17_materias_IS/1_mineria_de_datos/10_semana_miniproyecto3/1_algoritmo_nai

# dataset reading
tweets_de_trump <- read.csv("tweets_donald_trump.csv")
str(tweets_de_trump)
```

```
## 'data.frame':    204 obs. of  2 variables:
##  $ type : chr  "N" "N" "N" "P" ...
##  $ tweet: chr  "And strike and strike and strike and strike and strike! Aama lama rama lama bimbam!"
```

We transform the type element to factor

```
tweets_de_trump$type <- factor(tweets_de_trump$type)
str(tweets_de_trump)
```

```
## 'data.frame':    204 obs. of  2 variables:
##  $ type : Factor w/ 2 levels "N","P": 1 1 1 2 2 2 2 2 2 1 ...
##  $ tweet: chr  "And strike and strike and strike and strike and strike! Aama lama rama lama bimbam!"
```

```
table(tweets_de_trump$type)
```

```
##
##   N   P
## 132  72
```

DATA PROCESSING STAGE

Cleaning and standardization of text data

- we remove the numbers and punctuation

- we remove uninteresting words like and, but and or

- we decompose the sentences into individual words

package tm => remove numbers, punctuacion, uninteresting words as and, but and or, etc

```
# install.packages("tm")
library(tm)
```

```
## Loading required package: NLP
```

Corpus: Creation of text documents

```
# we create a corpus
tweets_corpus <- VCorpus( VectorSource(tweets_de_trump$tweet))
# we examine the corpus (Text document)
print(tweets_corpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 204
```

we get a summary of the corpus

```
inspect(tweets_corpus[ 1: 4])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 4
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 83
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 122
##
## [[3]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 108
##
## [[4]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 209
```

we see a single document

```
as.character(tweets_corpus[[1]])
```

```
## [1] "And strike and strike and strike and strike and strike! Aama lama rama lama bimbam!"
```

we see multiple documents with lapply()

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

```
## $`1`
## [1] "And strike and strike and strike and strike and strike! Aama lama rama lama bimbam!"
##
## $`2`
## [1] "EVERY American ought devote the time to listen to this 51 minute radio interview: How Trump is
```

we clean the data

we transform all tweets to lowercase with the function tolower()

```
tweets_corpus_clean <- tm::tm_map(tweets_corpus, content_transformer(tolower))

## we compare the two corpus
as.character(tweets_corpus[[3]])
```

```
## [1] "People are really tired of the whining. There's no conspiracy, no matter how much you try to co
```

```
as.character((tweets_corpus_clean[[3]]))
```

```
## [1] "people are really tired of the whining. there's no conspiracy, no matter how much you try to co
```

now we delete the numbers of the tweets

```
tweets_corpus_clean <- tm::tm_map(tweets_corpus_clean, removeNumbers)
```

we eliminate stop words(uninteresting words like and, but and or)

```
tweets_corpus_clean <- tm::tm_map(tweets_corpus_clean, removeWords, stopwords())
```

we remove the punctuation

```
tweets_corpus_clean <- tm::tm_map(tweets_corpus_clean, removePunctuation)
```

we define a function that replaces the punctuation

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

we do stemming (take words as playing, played and plays and transforms them to their base form => play)

```
# install.packages("SnowballC")
library(SnowballC)
wordStem(c("learn", "learned", "learning", "learns"))
```

```
## [1] "learn" "learn" "learn" "learn"
```

we apply the wordStem() function to all documents

```
tweets_corpus_clean <- tm::tm_map(tweets_corpus_clean, stemDocument)
as.character(tweets_corpus[[18]])
```

```
## [1] "Problem was our elections weren't secure. YOU refused to take any real action. The people then
```

```
as.character((tweets_corpus_clean[[18]]))
```

```
## [1] "problem elect secur refus take real action peopl took upon secur elect state get cheat time btw
```

we remove blank spaces

```
tweets_corpus_clean <- tm::tm_map(tweets_corpus_clean, stripWhitespace)

# data already cleaned
lapply(tweets_corpus[ 1: 3], as.character)
```

```
## $`1`
## [1] "And strike and strike and strike and strike and strike! Aama lama rama lama bimbam!"
##
## $`2`
## [1] "EVERY American ought devote the time to listen to this 51 minute radio interview: How Trump is
##
## $`3`
## [1] "People are really tired of the whining. There's no conspiracy, no matter how much you try to co
```

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

```
## $`1`
## [1] "strike strike strike strike strike aama lama rama lama bimbam"
##
## $`2`
## [1] "everi american devot time listen minut radio interview trump practic fascist polit"
##
## $`3`
## [1] "peopl realli tire whine conspiraci matter much tri conjur one"
```

DATA PREPARATION: Splitting text documents into words

we create a matrix => Document Term Matrix (DTM)

Each column represents a word and the intersection tells us if the word appears or not in the tweet

```
tweets_dtm <- DocumentTermMatrix(tweets_corpus_clean)
```

example of how we can create the DTM and clean the data at the same time

```
tweets_dtm_2 <- DocumentTermMatrix(tweets_corpus, control = list(
  tolower =  TRUE,
  removeNumbers = TRUE,
  stopwords = TRUE,
  removePunctuation = TRUE,
  stemming = TRUE
))

tweets_dtm_2
```

```
## <<DocumentTermMatrix (documents: 204, terms: 1065)>>
## Non-/sparse entries: 2364/214896
## Sparsity           : 99%
## Maximal term length: 20
## Weighting          : term frequency (tf)
```

We´ll divide the data into two porcions: 75 percent for training and 25 percent for testing

```
# SEPARAMOS LOS DATOS, EN DATOS DE ENTRENAMIENTO Y DATOS DE PRUEBA
# 75% para entrenamiento y 25% para prueba
tweets_dtm_train <- tweets_dtm[ 1: 153, ]
tweets_dtm_test <- tweets_dtm[ 154: 204, ]
tweets_train_labels <- tweets_de_trump[ 1: 153, ]$type  # variable a predecir y usamos el dataset origi
tweets_test_labels <- tweets_de_trump[ 154: 204, ]$type
```

We review the proportion of our 2 datasets; the training and the test.

```
prop.table( table(tweets_train_labels))
```

```
## tweets_train_labels
##         N         P
## 0.6078431 0.3921569
```

```
prop.table(table(tweets_test_labels))
```

```
## tweets_test_labels
##         N         P
## 0.7647059 0.2352941
```

we find frequent words

```
tweets_freq_words <- findFreqTerms( tweets_dtm_train, 5)
str(tweets_freq_words)
```

```
##  chr [1:76] "ago" "allow" "america" "american" "anoth" "ballot" "believ" ...
```

we filter our frequent terms in our matrix with training data and test data.

The white space indicates all lines.

```
tweets_dtm_freq_train <- tweets_dtm_train[ , tweets_freq_words]
tweets_dtm_freq_test <- tweets_dtm_test[ , tweets_freq_words]
```

The Naive Bayes classifier needs categorical data

function that converts values to categories

```
convert_counts <- function(x){x <- ifelse(x > 0, "Yes", "No")}
tweets_train <- apply(tweets_dtm_freq_train, MARGIN = 2, convert_counts)
tweets_test <- apply(tweets_dtm_freq_test, MARGIN = 2, convert_counts)
```

» APPLYING NAIVE BAYES

Step 3.- Train a data model

```
#install.packages("e1071")
library("e1071")

# La función naiveBayes() solo espera los datos de entrenamiento y la clase.
# construimos un modelo

 # le pasamos los datos de entrenamiento y etiquetas de entrenamiento
tweets_classifier <- naiveBayes(tweets_train, tweets_train_labels)
```

step 4.- Evaluate the performance of the model

we evaluate the model with our test data

```
tweets_test_pred <- predict(tweets_classifier, tweets_test) # >> le pasamos el modelo y los datos de pr
```

now we compare the predictions with the actual values

we compare what the algorithm said with reality

```
# install.packages("gmodels")
library(gmodels)
CrossTable( tweets_test_pred, tweets_test_labels, prop.chisq = FALSE, prop.t = FALSE, dnn = c(' predicte
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  51
##
##
##              | actual
##    predicted |         N |         P | Row Total |
## -------------|-----------|-----------|-----------|
##            N |        35 |        10 |        45 |
##              |     0.778 |     0.222 |     0.882 |
##              |     0.897 |     0.833 |           |
## -------------|-----------|-----------|-----------|
##            P |         4 |         2 |         6 |
##              |     0.667 |     0.333 |     0.118 |
##              |     0.103 |     0.167 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        39 |        12 |        51 |
```

```
##              |     0.765 |     0.235 |           |
## -------------|-----------|-----------|-----------|
##
##
```

SIMPLE OUT » we only pass predictions and test labels

```
table(tweets_test_pred, tweets_test_labels)
```

```
##                 tweets_test_labels
## tweets_test_pred  N  P
##                N 35 10
##                P  4  2
```

Step 5. Improving the performance model with laplace

```
tweets_classifier_2 <- naiveBayes(tweets_train, tweets_train_labels, laplace = 1)
tweets_test_pred_2 <- predict(tweets_classifier_2, tweets_test)

CrossTable(tweets_test_pred_2, tweets_test_labels, prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE, c
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  51
##
##
##              | actual
##    predicted |         N |         P | Row Total |
## -------------|-----------|-----------|-----------|
##            N |        20 |         5 |        25 |
##              |     0.513 |     0.417 |           |
## -------------|-----------|-----------|-----------|
##            P |        19 |         7 |        26 |
##              |     0.487 |     0.583 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        39 |        12 |        51 |
##              |     0.765 |     0.235 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```r
table(tweets_test_pred, tweets_test_labels)
```

```
##                 tweets_test_labels
## tweets_test_pred  N  P
##                N 35 10
##                P  4  2
```