Opgave 4

Jesse Piebenga

# Data

Data voor deze opgave: twts.csv (documentatie: twts.txt).

# Inleveren

Inleveren UITSLUITEND via Blackboard. Uiterste datum van inleveren: 21-3-2021 Inleveren UITSLUITEND in formaat .PDF, .DOC of .DOCX. Je kan, als je met RStudio werkt deze .Rmd file als basis gebruiken. Gebruik de “Knit” functie om pdf of docx te genereren. Zet je naam in het document bij “author” hierboven..

# Inleiding

De opgave gaat over het analyseren van tweets. De vraag is om vast te stellen of een tweet een positieve dan wel een negatieve lading heeft.

Bij deze opgave begin je met ruwe data. In de data staan een aantal variabelen die je waarschijnlijk beter niet kan gebruiken. Het kan zijn dat een aantal tweets meerdere malen voorkomt. (Misschien is het beter om deze teksten eerst te ontdubbelen, misschien maakt dat ook niet zo veel uit). Het is aan jou om deze data zodanig te behandelen dat je de analyse goed kan uitvoeren. De afhankelijke variabele is “Sentiment”.

De vraag is om een zo goed mogelijk CART-tree model te maken om uit de tekst van een tweet op te maken of het een positieve of negatieve lading heeft. De methode (text analytics) is uitgelegd op het college en is ook te vinden in de video’s “Turning Tweets into Knowledge”.

Bij deze opgave wordt van je verwacht dat je zelf de juiste stappen zet om tot het gewenste eindresultaat te komen. Probeer zo goed mogelijk uit te leggen WAAROM je bepaalde stappen zet.

# Vraag 1

Maak een zo optimaal mogelijk CART-tree model om uit de tekst van een tweet het (positieve/negatieve) sentiment te halen.

Let op: de Sentiment-variabele heeft DRIE verschillende waarden. Maak hiervan eerst een factor met slechts TWEE categorieen (positief/negatief). (Om te zorgen dat het een factor wordt kan je de functie as.factor gebruiken).

Gebruik daarvoor de tekst-analyse methodiek uit de video’s. Zorg er ook voor dat je de data splitst in een trainingsset en een testset, zodat je de validatie goed kan uitvoeren.

In je antwoorddocument verwacht ik in elk geval:

* Alle R-commando’s
* Relevant commentaar zodat het duidelijk is wat je doet en waarom je het doet
* Alle relevante R-uitvoer (bijvoorbeeld confusion matrices)
* Alle relevante R-plots
* Niet alleen het opstellen van een model (op basis training data) maar ook validatie van het model (op basis van testdata). Validatie bestaat in elk geval uit confusion matrix + accuracy + sensitivity + specificity en een ROC-curve, plus een argumentatie over hoe goed het model de betreffende data kan voorspellen.
* Een goed onderbouwde conclusie

library(tm)

## Loading required package: NLP

library(SnowballC)  
  
library(caret)

## Warning: package 'caret' was built under R version 4.0.4

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.4

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':  
##   
## annotate

library(e1071)

## Warning: package 'e1071' was built under R version 4.0.4

#read data from csv file  
twts = read.csv("twts.csv", stringsAsFactors = FALSE)  
#extract only the needed data  
twts = subset(  
 twts,  
 select = c(Sentiment, Tweet)  
)  
  
#add new factor Negative (TRUE or FALSE)  
twts$Negative = as.factor(twts$Sentiment < 2)  
  
#-------------------------------------------------------------------------------  
#Pre-processing  
  
#load tweets into the Corpus format (collection of documents)  
# making it easy to apply modifications to the provided text  
corpus = Corpus(VectorSource(twts$Tweet))  
#set all characters to lower  
corpus = tm\_map(corpus, tolower)

## Warning in tm\_map.SimpleCorpus(corpus, tolower): transformation drops documents

#remove all punctuation  
corpus = tm\_map(corpus, removePunctuation)

## Warning in tm\_map.SimpleCorpus(corpus, removePunctuation): transformation drops  
## documents

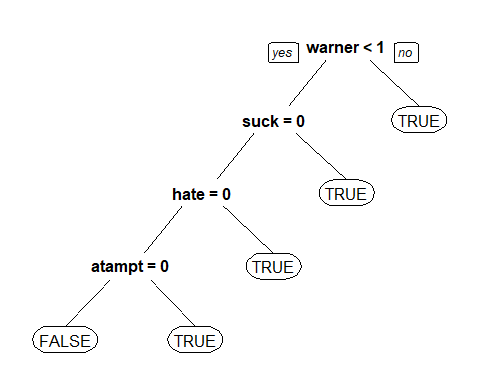
#remove all stop-words, these are not needed in our model  
corpus = tm\_map(corpus, removeWords, c(stopwords("english")))

## Warning in tm\_map.SimpleCorpus(corpus, removeWords, c(stopwords("english"))):  
## transformation drops documents

#stem all words  
corpus = tm\_map(corpus, stemDocument)

## Warning in tm\_map.SimpleCorpus(corpus, stemDocument): transformation drops  
## documents

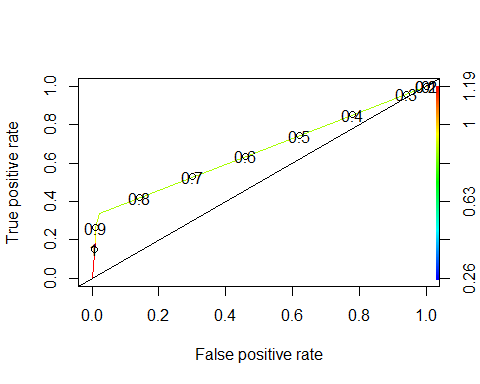
#-------------------------------------------------------------------------------  
#Bag of words  
  
#get frequencies of all the words in our corpus  
frequencies = DocumentTermMatrix(corpus)  
  
#remove rare words  
sparse = removeSparseTerms(frequencies, 0.995)  
  
#make sparse into a data frame  
tweetSparse = as.data.frame(as.matrix(sparse))  
#correcting names , words that start with numbers will be changed  
colnames(tweetSparse) = make.names(colnames(tweetSparse))  
  
#add independent variable  
tweetSparse$Negative = twts$Negative  
  
  
#-------------------------------------------------------------------------------  
#Splitting data  
  
library(caTools)  
#seed for pseudo random generator  
set.seed(123)  
#splitting the data  
split = sample.split(tweetSparse$Negative, SplitRatio = 0.7)  
#training set  
trainsparse = subset(tweetSparse, split == TRUE)  
#testing set  
testsparse = subset(tweetSparse, split == FALSE)  
  
#-------------------------------------------------------------------------------  
#CART-Tree  
  
library(rpart)  
library(rpart.plot)  
  
#create CART-Tree  
tweetCART = rpart(Negative ~ ., data = trainsparse, method ="class")   
#plot Cart-Tree  
prp(tweetCART)



#prediction from our model applied to our test-set  
predictCart = predict(tweetCART, newdata = testsparse, type="class")  
  
#Confusion Matrix  
print(confusionMatrix(testsparse$Negative, predictCart))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction FALSE TRUE  
## FALSE 94 2  
## TRUE 35 18  
##   
## Accuracy : 0.7517   
## 95% CI : (0.6743, 0.8187)  
## No Information Rate : 0.8658   
## P-Value [Acc > NIR] : 0.9999   
##   
## Kappa : 0.3704   
##   
## Mcnemar's Test P-Value : 1.435e-07   
##   
## Sensitivity : 0.7287   
## Specificity : 0.9000   
## Pos Pred Value : 0.9792   
## Neg Pred Value : 0.3396   
## Prevalence : 0.8658   
## Detection Rate : 0.6309   
## Detection Prevalence : 0.6443   
## Balanced Accuracy : 0.8143   
##   
## 'Positive' Class : FALSE   
##

#-------------------------------------------------------------------------------  
#ROC-Curve  
  
library(ROCR)  
#create ROC predict  
predictTreeROC = predict(tweetCART,newdata = testsparse)  
  
ROCRpred = prediction(predictTreeROC[,2],testsparse$Negative)  
perf = performance(ROCRpred, "tpr", "fpr")  
#Plot the ROC\_Curve  
plot(perf,  
 colorize = TRUE,  
 print.cutoffs.at = seq(0, 1, 0.1))  
#Addition of linear line  
abline(0, 1)



#Calculate Area under the Curve  
AUC = as.numeric(performance(ROCRpred, "auc")@y.values)  
print(AUC)

## [1] 0.6599843

#[1] 0.659984

# Vraag 2

Bij de eerste vraag heb je de 3 waarden voor Sentiment omgezet in 2 waarden voor de tekst-analyse. Je zou ook een CART-tree kunnen maken met daarbij alle drie waarden van Sentiment. Doe dit en geef het resultaat.

Gebruik cross-validation om de best mogelijke cp-parameter vast te stellen hiervoor.

De confusion-matrix is nu 3 bij 3. Geef aan wat dat voor consequenties heeft voor je validatie.

In je antwoorddocument verwacht ik in elk geval:

* Alle R-commando’s
* Relevant commentaar zodat het duidelijk is wat je doet en waarom je het doet
* Alle relevante R-uitvoer (bijvoorbeeld confusion matrices)
* Alle relevante R-plots
* Je antwoord op de vraag in de laatste zin van vraag 2 (over de 3x3 matrix)
* Een goed onderbouwde conclusie

library(tm)  
library(SnowballC)  
  
library(caret)  
library(e1071)  
  
#read data from csv file  
twts = read.csv("twts.csv", stringsAsFactors = FALSE)  
#extract only the needed data  
twts = subset(  
 twts,  
 select = c(Sentiment, Tweet)  
)  
  
twts$Sentiment = as.factor(twts$Sentiment)  
#-------------------------------------------------------------------------------  
#Pre-processing  
  
#load tweets into the Corpus format (collection of documents)  
# making it easy to apply modifications to the provided text  
corpus = Corpus(VectorSource(twts$Tweet))  
#set all characters to lower  
corpus = tm\_map(corpus, tolower)

## Warning in tm\_map.SimpleCorpus(corpus, tolower): transformation drops documents

#remove all punctuation  
corpus = tm\_map(corpus, removePunctuation)

## Warning in tm\_map.SimpleCorpus(corpus, removePunctuation): transformation drops  
## documents

#remove all stop-words, these are not needed in our model  
corpus = tm\_map(corpus, removeWords, c(stopwords("english")))

## Warning in tm\_map.SimpleCorpus(corpus, removeWords, c(stopwords("english"))):  
## transformation drops documents

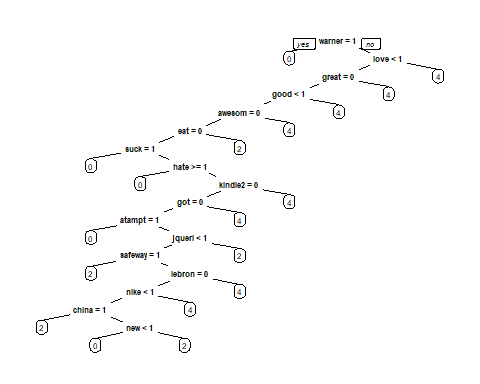
#stem all words  
corpus = tm\_map(corpus, stemDocument)

## Warning in tm\_map.SimpleCorpus(corpus, stemDocument): transformation drops  
## documents

#-------------------------------------------------------------------------------  
#Bag of words  
  
#get frequencies of all the words in our corpus  
frequencies = DocumentTermMatrix(corpus)  
  
#remove rare words  
sparse = removeSparseTerms(frequencies, 0.995)  
  
#make sparse into a data frame  
tweetSparse = as.data.frame(as.matrix(sparse))  
#correcting names , words that start with numbers will be changed  
colnames(tweetSparse) = make.names(colnames(tweetSparse))  
  
#add independent variable  
tweetSparse$Sentiment = twts$Sentiment  
  
  
#-------------------------------------------------------------------------------  
#Splitting data  
  
library(caTools)  
#seed for pseudo random generator  
set.seed(123)  
#splitting the data  
split = sample.split(tweetSparse$Sentiment, SplitRatio = 0.7)  
#training set  
trainsparse = subset(tweetSparse, split == TRUE)  
#testing set  
testsparse = subset(tweetSparse, split == FALSE)  
  
#-------------------------------------------------------------------------------  
#CART-Tree  
  
library(rpart)  
library(rpart.plot)  
  
#use k-fold cross validation  
fitControl = trainControl(method = "cv", number = 10)  
cartGrid = expand.grid(.cp=(1:50)\*0.01)  
cv = train(Sentiment~. , data = trainsparse, method="rpart",trControl = fitControl, tuneGrid = cartGrid)  
  
print(cv)

## CART   
##   
## 348 samples  
## 316 predictors  
## 3 classes: '0', '2', '4'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 315, 313, 313, 313, 314, 312, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.01 0.5196902 0.263630547  
## 0.02 0.4392870 0.136588210  
## 0.03 0.4283393 0.101369292  
## 0.04 0.4313696 0.105354578  
## 0.05 0.4313696 0.105354578  
## 0.06 0.4313696 0.105354578  
## 0.07 0.4313696 0.105354578  
## 0.08 0.4313696 0.105354578  
## 0.09 0.4313696 0.105354578  
## 0.10 0.3932871 0.045016848  
## 0.11 0.3678902 0.004545455  
## 0.12 0.3649491 0.000000000  
## 0.13 0.3649491 0.000000000  
## 0.14 0.3649491 0.000000000  
## 0.15 0.3649491 0.000000000  
## 0.16 0.3649491 0.000000000  
## 0.17 0.3649491 0.000000000  
## 0.18 0.3649491 0.000000000  
## 0.19 0.3649491 0.000000000  
## 0.20 0.3649491 0.000000000  
## 0.21 0.3649491 0.000000000  
## 0.22 0.3649491 0.000000000  
## 0.23 0.3649491 0.000000000  
## 0.24 0.3649491 0.000000000  
## 0.25 0.3649491 0.000000000  
## 0.26 0.3649491 0.000000000  
## 0.27 0.3649491 0.000000000  
## 0.28 0.3649491 0.000000000  
## 0.29 0.3649491 0.000000000  
## 0.30 0.3649491 0.000000000  
## 0.31 0.3649491 0.000000000  
## 0.32 0.3649491 0.000000000  
## 0.33 0.3649491 0.000000000  
## 0.34 0.3649491 0.000000000  
## 0.35 0.3649491 0.000000000  
## 0.36 0.3649491 0.000000000  
## 0.37 0.3649491 0.000000000  
## 0.38 0.3649491 0.000000000  
## 0.39 0.3649491 0.000000000  
## 0.40 0.3649491 0.000000000  
## 0.41 0.3649491 0.000000000  
## 0.42 0.3649491 0.000000000  
## 0.43 0.3649491 0.000000000  
## 0.44 0.3649491 0.000000000  
## 0.45 0.3649491 0.000000000  
## 0.46 0.3649491 0.000000000  
## 0.47 0.3649491 0.000000000  
## 0.48 0.3649491 0.000000000  
## 0.49 0.3649491 0.000000000  
## 0.50 0.3649491 0.000000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.01.

#create CART-Tree  
tweetCART = rpart(Sentiment ~ ., data = trainsparse, method ="class", control = rpart.control(cp=0.01))   
#plot Cart-Tree  
prp(tweetCART)



#prediction from our model applied to our test-set  
predictCart = predict(tweetCART, newdata = testsparse, type="class")  
#Confusion Matrix  
print(confusionMatrix(testsparse$Sentiment, predictCart))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 2 4  
## 0 40 1 12  
## 2 22 16 4  
## 4 25 4 26  
##   
## Overall Statistics  
##   
## Accuracy : 0.5467   
## 95% CI : (0.4634, 0.628)  
## No Information Rate : 0.58   
## P-Value [Acc > NIR] : 0.8187   
##   
## Kappa : 0.306   
##   
## Mcnemar's Test P-Value : 2.828e-05   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 2 Class: 4  
## Sensitivity 0.4598 0.7619 0.6190  
## Specificity 0.7937 0.7984 0.7315  
## Pos Pred Value 0.7547 0.3810 0.4727  
## Neg Pred Value 0.5155 0.9537 0.8316  
## Prevalence 0.5800 0.1400 0.2800  
## Detection Rate 0.2667 0.1067 0.1733  
## Detection Prevalence 0.3533 0.2800 0.3667  
## Balanced Accuracy 0.6267 0.7802 0.6753

# Eindbeoordeling Analytics

Als je alle vier opgaven hebt ingeleverd en deze zijn voldoende beoordeeld wordt je uitgenodigd om de resultaten van deze opgave in een assessment te bespreken met de docent. (Het assessment duurt ongeveer 8 minuten). Het eindcijfer van het vak Analytics wordt bepaald in het assessment. Zie voor meer informatie het document “Afsluitend assessment Analytics” op Blackboard.