```
# Titol: Fake news analysis and classification
# Exploratory Analysis and fake news classification on Buzzfeed News
# Web: https://www.kaggle.com/code/kumudchauhan/fake-news-analysis-and-
classification/report
#
# 1.-COMENTARIS SOBRE LES DADES-
# El dataset The Buzzfeed news dataset inclou una mostra de noticies publicades a
Facebook
# per 9 agencies de noticies entre les dates 19-23 i 26-27 de Septembre de 2016.

# El dataset Buzzfeed news consisteix en 2 datasets: un dataset de noticies falses i un
dataset de
# noticies reals en 2 fitxers csv. Alguns dels atributs d'aquests datasets són:

#id: identificatiu de la noticia (tant al dataset de noticies falses com al de noticies
reals.
#title : titular de la noticia (real o inventada).
#text : cos de l'article.
#source: nom de l'autor de l'article.
#images: imatges vinculades a l'article.
#movies: videos vinculats a l'article.

# Importació de Packages o Libraries
library(tm) # for NLP
library(plyr) #  for pre-processing
library(tidyverse) # for pre-processing and visualisation
library(reshape2) # for melt function
library(e1071) # for Naive Bayes classifier
library(glmnet) # for Logistic Regression classifier
library(randomForest) # for Random Forest classifier
library(SnowballC) # implements Porter's word stemming algorithm

# 1.1- Carrega de datasets Buzzfeed :
buzzfeed_real <-
read_csv('C:\\Users\\Alumne_mati1\\BigData\\input\\BuzzFeed_real_news_content.csv')
buzzfeed_real

buzzfeed_fake <-
read_csv('C:\\Users\\Alumne_mati1\\BigData\\input\\BuzzFeed_fake_news_content.csv')
buzzfeed_fake

# 1.2- Pre- processament
# fusió i esborrament de dataframes antics
buzzfeed_df = rbind(buzzfeed_real, buzzfeed_fake)

# afegim una nova columna "type" per indicar si es tracta d'un article real o inventat
buzzfeed_df$type <- sapply(strsplit(buzzfeed_df$id, "_"), head,  1)

# Dimensions del dataset
dim(buzzfeed_df)

# Summary del dataset
summary(buzzfeed_df)

row.names(buzzfeed_df)

# selecció de columnes del dataframe per l'analisi
buzzfeed_df <- buzzfeed_df %>%
dplyr::select(c("id","title","text","source","type","images","movies"))

# Les columnes movies i images tindran valor 1 o valor 0 en funció de si hi ha o no
elements d'aquestes categories.
buzzfeed_df$movies<- ifelse(is.na(buzzfeed_df$movies) , 0, 1)
buzzfeed_df$images<- ifelse(is.na(buzzfeed_df$images) , 0, 1)

# La web "addictinginfo.org" és una font de noticies amb una url diferent
# Combinació de totes les fonts de "addictinginfo.org" per analitzar.
buzzfeed_fake$source <- gsub("www.addic|author.addic", "addic",buzzfeed_fake$source)
buzzfeed_real$source <- gsub("www.addic|author.addic", "addic",buzzfeed_real$source)
```

```
# 2.- ESTUDI PREVI A L'ANALISI-
# 2.1- Analisi de les noticies Reals i les noticies Inventades
# 2.1.1- Fonts que publiquen noticies reals
buzzfeed_real$source <- with(buzzfeed_real, reorder(source, source, function(x)
length(x)))
ggplot(data = buzzfeed_real) +
  ggtitle("source count of real news in Buzzfeed") +
  geom_bar(aes(x= source),fill = "green") + coord_flip()

# 2.1.2- Fonts que publiquen noticies inventades
buzzfeed_fake$source <- with(buzzfeed_fake,reorder(source, source, function(x)
length(x)))
ggplot(buzzfeed_fake) +
  ggtitle("source count of fake news in Buzzfeed") +
  geom_bar(aes(x=source),fill = "red") + coord_flip()

# 2.1.3- Fonts comunes de noticies reals i noticies inventades
common_source <- intersect(buzzfeed_real$source,buzzfeed_fake$source)
source_type_counts = table(buzzfeed_df$source, buzzfeed_df$type)

# 2.1.4- Bar chart de les fonts
ggplot(buzzfeed_df[which(buzzfeed_df$source %in% common_source),]) +
  geom_bar(aes(x = source,fill = type),position = "dodge") + coord_flip() +
  ggtitle("common source of real and fake news in Buzzfeed")

# 2.1.5- Bar chart de la columna "movies"
ggplot(buzzfeed_df) +
  geom_bar(aes(x= factor(movies), fill = type),position = "dodge")  +
  xlab("Movies linked to news") + ylab("counts") +
  theme() + ggtitle("News category wise movies")

# 2.1.6- Bar chart de la columna "images"
ggplot(buzzfeed_df) +
  geom_bar(aes(x= as.factor(images), fill = type),position = "dodge")+
  xlab("Images in news") + ylab("counts") +
  theme() + ggtitle("News category wise images")

# 2.1.7- Neteja de la columna "text"
clean_text <- function(x){
  gsub("…|★|-|‹|”|‟|‘|’", " ", x)
}

# 2.1.8- Funció per esborrar un conjunt de caracters o simbols
preprocess_corpus <- function(corpus){
  # Convert the text to lower case
  corpus <- tm_map(corpus, content_transformer(tolower))
  # Remove numbers
  corpus <- tm_map(corpus, removeNumbers)
  # Remove punctuations
  corpus <- tm_map(corpus, removePunctuation)
  # Remove special characters from text
  corpus <- tm_map(corpus, clean_text)
  # Remove english common stopwords
  corpus <- tm_map(corpus, removeWords, stopwords("english"))
  # Remove name of newspapers from the corpus
  corpus <- tm_map(corpus, removeWords, c("eagle rising","freedom daily"))
  # 'stem' words to root words
  corpus <- tm_map(corpus,stemDocument)
  # Eliminate extra white spaces
  corpus <- tm_map(corpus, stripWhitespace)
  return (corpus)
}

# 2.2.1- Funció per comptabilitzar la frequencia de determinades paraules als articles
reals o inventats
find_category_representative_words_using_chi_sq <- function(dtf_matrix, categories,
top_n=20){
  dtm_df <- data.frame(dtf_matrix)
```

```
  # find top features using chi-sq test
  chi2vals <- apply(dtf_matrix, 2, function(x){
    chisq.test(as.numeric(x), categories)$statistic
  })
  features_subset <- names(sort(chi2vals, decreasing = TRUE))[1:top_n]

  # Compute term frequency for top terms in both categories
  dtm_df$NewsType <- categories
  cat_freq_df <- dtm_df %>% group_by(NewsType) %>% summarise_each(funs(sum))
  top_words_freq <- cat_freq_df[, c(features_subset, "NewsType")]
  return (top_words_freq)
}

#2.2.2- Analisi de title corpus a Buzzfeed
title_corpus <- Corpus(VectorSource(buzzfeed_df$title))
# Conversió de title corpus a document term matrix
title_dtm <- DocumentTermMatrix(preprocess_corpus(title_corpus))
title_dtm_matrix <- as.matrix(title_dtm)

#2.2.3- Recerca de 20 paraules top a la columna "title" per ambdues categories: real o
inventat
title_top_words_freq <-
find_category_representative_words_using_chi_sq(title_dtm_matrix,buzzfeed_df$type,20)
# Visualitzar la frequencia de 20 paraules top 'discriminatory' a la columna "title"
ggplot(melt(title_top_words_freq),aes(x =variable, y =value,fill = NewsType)) +
geom_col(position = "dodge") + coord_flip() + xlab("Top 20 words") + ylab("Term
Frequency of words") +
  theme() + ggtitle("Most discriminatory words in the title of news")

#2.2.4- Analisi sobre el cos del articles de Buzzfeed (unigrames)

# Corpus de Buzzfeed text
body_corpus <- Corpus(VectorSource(buzzfeed_df$text))
# Conversió de body corpus a document term matrix
body_dtm <- DocumentTermMatrix(preprocess_corpus(body_corpus))
body_dtm_matrix <- as.matrix(body_dtm)
# Recerca de 30 paraules top al cos a ambdues categories finding top 30 words in the
news body
body_top_words_freq <-
find_category_representative_words_using_chi_sq(body_dtm_matrix,buzzfeed_df$type,30)
# Visualització de la frequencia de 30 paraules 'discriminatory' al cos de l'article
ggplot(melt(body_top_words_freq),aes(x =variable, y=value,fill = NewsType)) +
  geom_col(position = "dodge") + coord_flip() + xlab("Top 30 words") + ylab("Term
Frequency of words") +
  theme() + ggtitle("Most discriminatory words in the body of news article")

#2.2.5- Analisi de la longitud de la columna 'title'
# longitud del title per l'histograma
title_length <- rowSums(title_dtm_matrix)
# longitud de la columna 'title' del dataframe amb categories
tl_df <- data.frame(title_length, buzzfeed_df$type)

# perform t-test
t.test(tl_df[tl_df$buzzfeed_df.type == "Real",]$title_length,
tl_df[tl_df$buzzfeed_df.type == "Fake",]$title_length)

# Visualització del histograma de la longitud de 'title'
ggplot(tl_df ,aes(x = title_length, fill = buzzfeed_df.type)) +
  geom_density(alpha=0.5) +
  guides(fill=guide_legend(title="News type")) +
  xlab("Title length") + ylab("Density") + theme() +
  ggtitle("Density distribuiton of title length for real and fake news")
#2.2.6- Analisi de Bigrames al cos dels articles.

# funció per tokenitzar bigrames
BigramTokenizer <-
  function(x)
    unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)
# corpus per bigrames
```

```
corpus <- VCorpus(VectorSource(buzzfeed_df$text))
# corpus to document term matrix of bigrams
bigram_matrix <- DocumentTermMatrix(corpus, control = list(tokenize = BigramTokenizer))
# sort frequency of bigrams in decreasing order to give high frequency phrases
bigram_freq <- sort(colSums(as.matrix(bigram_matrix)), decreasing=TRUE)

# funció per buscar top bigrames
find_top_bigram <- function(bigrams, top_n){
  top_bigram_list <- c()
  for(bigram in bigrams){
    unigrams <- strsplit(bigram," ")
    if(!(unigrams[[1]][1] %in% stopwords("en") | unigrams[[1]][2]  %in%
stopwords("en"))){
      top_bigram_list <- c(top_bigram_list, bigram)
    }
    if (length(top_bigram_list) ==top_n){
      break
    }
  }
  return (top_bigram_list)
}

features_subset <- find_top_bigram(names(bigram_freq), 20)
dtm_bigram_df <- data.frame(as.matrix(bigram_matrix[,
intersect(colnames(bigram_matrix),features_subset)]))
dtm_bigram_df$NewsType <- buzzfeed_df$type
cat_freq_bf_df <- dtm_bigram_df %>% group_by(NewsType) %>% summarise_each(funs(sum))

# plot high frequency bigrams in the body of news articles.
ggplot(melt(cat_freq_bf_df),aes(x =variable, y=value,fill = NewsType)) +
  geom_col(position = "dodge") + coord_flip() + xlab("bigrams") +
ylab("bigrams_frequency") +
  theme() + ggtitle("High frequency bigrams in the body of news article")


#3. JUSTIFICACIO DE LA TRIA DEL MODEL ESCOLLIT I ELS SEUS PARAMETRES
#Fake/Real news classification
#3.1- Train i test data split
#dividim les dades en un dataset de training i un dataset de test
#en una proporció de 75% pel dataset de training i 25% pel dataset test.
set.seed(123)
n_obs <- nrow(buzzfeed_df)
prop_split <- .75
training_index <- sample(1:n_obs, round(n_obs * prop_split))

#3.2- Detecció de noticies falses a la columna 'title' de l'article
inspect(title_dtm[100:105,100:105]) # 100% sparsity

# Esborrar sparse terms
sparse_title_dtm <- removeSparseTerms(title_dtm, .997) # 750 terms
sparse_title_dtm

title_dtm <- as.matrix(sparse_title_dtm)
# set train and test set for title dtm
y_true <- as.matrix(buzzfeed_df$type)
x_train <- title_dtm[training_index, ]
x_test <- title_dtm[-training_index, ]

#3.2.1- Naive Bayes Classifier com Base line model
nb_title <- naiveBayes(x=x_train , y=as.factor(y_true[training_index]))
predicted_rf_title <- predict(nb_title, x_test)
accuracy_nb_title <- sum(y_true[-training_index] == predicted_rf_title)/
length(predicted_rf_title)
accuracy_nb_title

#3.2.2- Logistic Regression Classifier
glm_fit_title <- glmnet(x_train , y_true[training_index], family = "binomial")
predicted_glm_title <- predict(glm_fit_title, x_test, type = "class")
accuracy_glm_title <- sum(y_true[-training_index] == predicted_glm_title)/
```

```
length(predicted_glm_title)
accuracy_glm_title

#3.2.3- Random Forest Classifier
set.seed(123)
rf_title <- randomForest(x=x_train, y=as.factor(y_true[training_index]),ntree = 50)
rf_title

predicted_rf_title <- predict(rf_title, newdata=x_test)
accuracy_rf_title <- sum(y_true[-training_index] == predicted_rf_title)/
length(predicted_rf_title)
accuracy_rf_title

#3.3- Detecció de noticies falses al cos de l'article
# Esborrar paraules no habituals
sparse_body_dtm <- removeSparseTerms(body_dtm, 0.97) # 1337 terms
sparse_body_dtm

body_dtm <- as.matrix(sparse_body_dtm)
# set train and test set for body dtm
y_true <- as.matrix(buzzfeed_df$type)
x_train_body <- body_dtm[training_index,]
x_test_body <- body_dtm[-training_index, ]

#3.3.1- Naive Bayes Classifier com Base line model
nb_body <- naiveBayes(x=x_train_body , y=as.factor(y_true[training_index]))
predicted_naive_body <- predict(nb_body, x_test_body)
accuracy_naive_body <- sum(y_true[-training_index] == predicted_naive_body)/
length(predicted_naive_body)
accuracy_naive_body

#3.3.2- Logistic Regression Classifier
glm_fit_body <- glmnet(x_train_body , y_true[training_index], family = "binomial")
predicted_glm_body <- predict(glm_fit_body, x_test_body, type = "class")
accuracy_glm_body <- sum(y_true[-training_index] == predicted_glm_body)/
length(predicted_glm_body)
accuracy_glm_body

#3.3.3- Random Forest Classifier
set.seed(123)
rf_body <- randomForest(x=x_train_body, y=as.factor(y_true[training_index]))
rf_body

predicted_rf_body <- predict(rf_body, newdata=x_test_body)
accuracy_rf_body <- sum(y_true[-training_index] == predicted_rf_body)/
length(predicted_rf_body)
accuracy_rf_body

#3.4-Detecció de notícies falses mitjançant paraules que apareixen al títol o al cos de
la notícia
#combinar les paraules del títol i del cos com a feature matrix
title_body_dtm <- body_dtm
common_features <- intersect(colnames(body_dtm), colnames(title_dtm))
title_body_dtm[,common_features] <- body_dtm[,common_features]+
title_dtm[,common_features]
title_only_features <- setdiff(colnames(title_dtm), colnames(body_dtm))
title_body_dtm <- cbind(title_body_dtm, title_dtm[,title_only_features])

#3.4.1- Naive Bayes Classifier com Base line model
nb_body_tb <- naiveBayes(x=title_body_dtm[training_index, ] ,
y=as.factor(y_true[training_index]))
predicted_nb_tb <- predict(nb_body, title_body_dtm[-training_index, ])
accuracy_nb_tb <- sum(y_true[-training_index] == predicted_nb_tb)/
length(predicted_nb_tb)
accuracy_nb_tb

#3.4.2- Logistic Regression Classifier
glm_fit_title_body <- glmnet(x=title_body_dtm[training_index, ] ,
y=y_true[training_index], family = "binomial")
```

```
predicted_glm_tb <- predict(glm_fit_title_body, title_body_dtm[-training_index, ], type
= "class")
accuracy_glm_tb <- sum(y_true[-training_index] == predicted_glm_tb)/
length(predicted_glm_tb)
accuracy_glm_tb

#3.4.3- Random Forest Classifier
set.seed(123)
rf_tb <- randomForest(x=title_body_dtm[training_index, ],
y=as.factor(y_true[training_index]))
predicted_rf_tb <- predict(rf_tb, newdata=title_body_dtm[-training_index, ])
accuracy_rf_tb <- sum(y_true[-training_index] == predicted_rf_tb)/
length(predicted_rf_tb)
accuracy_rf_tb

#4. VALORACIO DEL RESULTAT OBTINGUT
#Hem constatat que analitzar dades de text és una mica més dificultós que les dades
#numèriques. Hem realitzat les següents tasques en aquest projecte: Hem fet una anàlisi
#exploratòria detallada de dades sobre notícies reals i inventades del dataset buzzfeed
i #finalment, hem generat múltiples gràfics de totes les variables per a les dues
#categories de notícies.

#Hem analitzat els unigrames i els bigrames i vam obtenir algunes paraules
#i frases interessants que s'associen a notícies falses i s'inclouen al títol o al cos
#de la notícia. Tanmateix, reconeixem que algunes frases/bigrames haurien de ser
#netejades. Però, creiem que en aquest tipus d'anàlisi eliminar les paraules buides
#pot no ser una bona idea, ja que podríem perdre una mica d'informació.
#Hi ha algunes paraules i frases comuns que poden ser associades amb un tipus
particular
#d'informe de notícies i es poden utilitzar per manipular el llenguatge del títol
#o el cos de la notícia. Per això no hem netejat el text en l'anàlisi de bigrams.

#Construïm uns classificadors binaris que classifiquen notícies inventades i notícies
reals
#en base a termes (unigrames) apareix al títol, al cos o a tots dos
#de la notícia. Utilitzarem tres classificadors logístics diferents
#Regression, Random Forest, Naive-Bayes classifier per detectar notícies inventades.

#El Naive-Bayes classifier és el nostre model de referència i el Random Forest
#és el millor model per a aquest anàlisi amb resultats més precisos
#de classificació. No obstant això, la precisió del Logistic Regression classifier
#també millora amb la matriu de funcions combinada. L'últim Random Forest classifier
#aconsegueix una precisió màxima del 80% amb la matriu de característiques del títol
#combinada i el body dtm.

#Podem entrenar models utilitzant bigrams i altres funcions com les fonts,
#videos, imatges per comprovar l'efecte sobre la precisió dels models i analitzar
#com podem detectar les notícies inventades mitjançant altres funcions.
```