NLP project - Bartomeu Ramis

```
library(keras)
library(dplyr)
library(ggplot2)
library(tm)
library(corpus)
library(wordcloud)
require(quanteda)
require(quanteda.textmodels)
library(quanteda.textplots)
require(caret)
```

Definition of the problem and the data

This data set includes 23486 rows and 10 feature variables. Each row corresponds to a customer review, and includes the variables:

```
Clothing ID: Integer Categorical variable that refers to the specific piece being reviewed.
Age: Positive Integer variable of the reviewers age.
Title: String variable for the title of the review.
Review Text: String variable for the review body.
Rating: Positive Ordinal Integer variable for the product score granted by the customer from 1 Worst, t
Recommended IND: Binary variable stating where the customer recommends the product where 1 is recommend
Positive Feedback Count: Positive Integer documenting the number of other customers who found this revi
Division Name: Categorical name of the product high level division.
Department Name: Categorical name of the product department name.
Class Name: Categorical name of the product class name.
```

```
set.seed(222)
data = read.csv("Womens Clothing E-Commerce Reviews.csv", header=TRUE)
head(data)
```

```
##   X Clothing.ID Age                   Title
## 1 0         767  33
## 2 1        1080  34
## 3 2        1077  60 Some major design flaws
## 4 3        1049  50        My favorite buy!
## 5 4         847  47        Flattering shirt
## 6 5        1080  49 Not for the very petite
##
## 1
## 2
## 3 I had such high hopes for this dress and really wanted it to work for me. i initially ordered the p
## 4
```

```
## 5
## 6             I love tracy reese dresses, but this one is not for the very petite. i am just under 5
##   Rating Recommended.IND Positive.Feedback.Count  Division.Name Department.Name
## 1      4               1                       0      Initmates        Intimate
## 2      5               1                       4        General         Dresses
## 3      3               0                       0        General         Dresses
## 4      5               1                       0 General Petite         Bottoms
## 5      5               1                       6        General            Tops
## 6      2               0                       4        General         Dresses
##   Class.Name
## 1  Intimates
## 2    Dresses
## 3    Dresses
## 4      Pants
## 5    Blouses
## 6    Dresses
```

Objetive

Our objective with this data set, is to create a NLP model, that using the review text from a customer, can predict if that costumer was satisfied or not with it's purchase. In our case, we will use de Recommended value as an indicator of the happiness of the customer.

Data clearing

So, first of all, lets drop those columns that we wont be needing. - Clothing ID could be useful if we were interested in which clothes have better opinions, but in our case, we don't really care. - Age is another piece of information that, for our purpose, we don't need. - Rating would be extremely useful if we had not the recommend IND. However, we will keep it, just in case. - Positive feedback can also be omitted, given the fact that represents the opinion of other customers on the review, and gives no insight into the review's expressed opinions. - And finally, Division Name, Department Name and Class name wont bring any useful information for our objectives.

Also, we will change the names of some columns to be more intuitive and usable

```r
data = subset(data, select=c(X, Title, Review.Text, Rating, Recommended.IND))
names(data)[names(data) == "X"] <- "id"
names(data)[names(data) == "Review.Text"] <- "Text"
names(data)[names(data) == "Recommended.IND"] <- "Recommend"
summary(data)
```

```
##        id           Title               Text               Rating     
##  Min.   :    0   Length:23486       Length:23486       Min.   :1.000  
##  1st Qu.: 5871   Class :character   Class :character   1st Qu.:4.000  
##  Median :11742   Mode  :character   Mode  :character   Median :5.000  
##  Mean   :11742                                         Mean   :4.196  
##  3rd Qu.:17614                                         3rd Qu.:5.000  
##  Max.   :23485                                         Max.   :5.000  
##    Recommend     
##  Min.   :0.0000  
##  1st Qu.:1.0000  
##  Median :1.0000  
```

```
##  Mean   :0.8224
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

The title of the review has also some key words that will be quite helpful to determinate the "feelings" of the customer. So in order to simplify the learning process we will merge the titled and the review text in a new column named "all_text", and we will drop the "title" and "text" from our data set to make it more lightweight.

```
data$all_text <-paste(data$Title,data$Text, sep=" ")
data = subset(data, select=c(id, Rating, Recommend, all_text))
head(data)
```

```
##   id Rating Recommend
## 1  0      4         1
## 2  1      5         1
## 3  2      3         0
## 4  3      5         1
## 5  4      5         1
## 6  5      2         0
##
## 1
## 2
## 3 Some major design flaws I had such high hopes for this dress and really wanted it to work for me. 
## 4
## 5
## 6             Not for the very petite I love tracy reese dresses, but this one is not for the very p
```

Now, let's look for null values.

```
summary(data)
```

```
##        id            Rating        Recommend        all_text
##  Min.   :    0   Min.   :1.000   Min.   :0.0000   Length:23486
##  1st Qu.: 5871   1st Qu.:4.000   1st Qu.:1.0000   Class :character
##  Median :11742   Median :5.000   Median :1.0000   Mode  :character
##  Mean   :11742   Mean   :4.196   Mean   :0.8224
##  3rd Qu.:17614   3rd Qu.:5.000   3rd Qu.:1.0000
##  Max.   :23485   Max.   :5.000   Max.   :1.0000
```

```
cat("number of na/nan values: ", sum(is.na(data)),"\n")
```

```
## number of na/nan values:  0
```

```
cat("number of na/nan values for 'all_text' column: ",sum(is.na(data$all_text)), "\n")
```

```
## number of na/nan values for 'all_text' column:  0
```

```
cat("number of na/nan values for 'recommend' column: ",sum(is.na(data$Recommend)), "\n")
```

```
## number of na/nan values for 'recommend' column:  0
```

It seem that the data is already clean from missing values. Next let's look for outliers or non valid values.

```
cat("number of values bigger than 1 or smaller than 0 for Recommend: ", sum(data$Recommend > 1 | data$Re
```

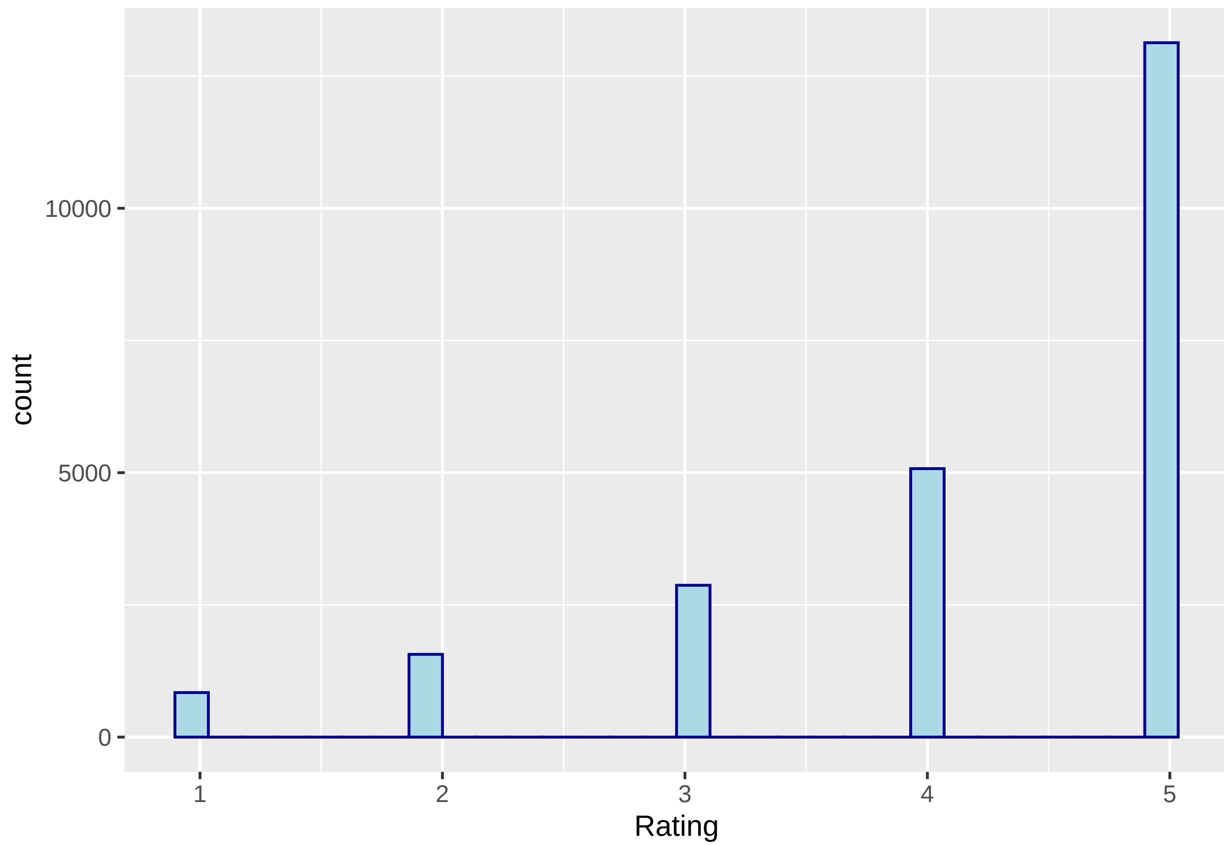
```
## number of values bigger than 1 or smaller than 0 for Recommend:  0
```

```
cat("number of values smaller than 1 and bigger than 0 for Recommend: ",sum(data$Recommend < 1 & data$Re
```

```
## number of values smaller than 1 and bigger than 0 for Recommend:  0
```

```
ggplot(data, aes(x=Rating)) + geom_histogram(color="darkblue", fill="lightblue")
```

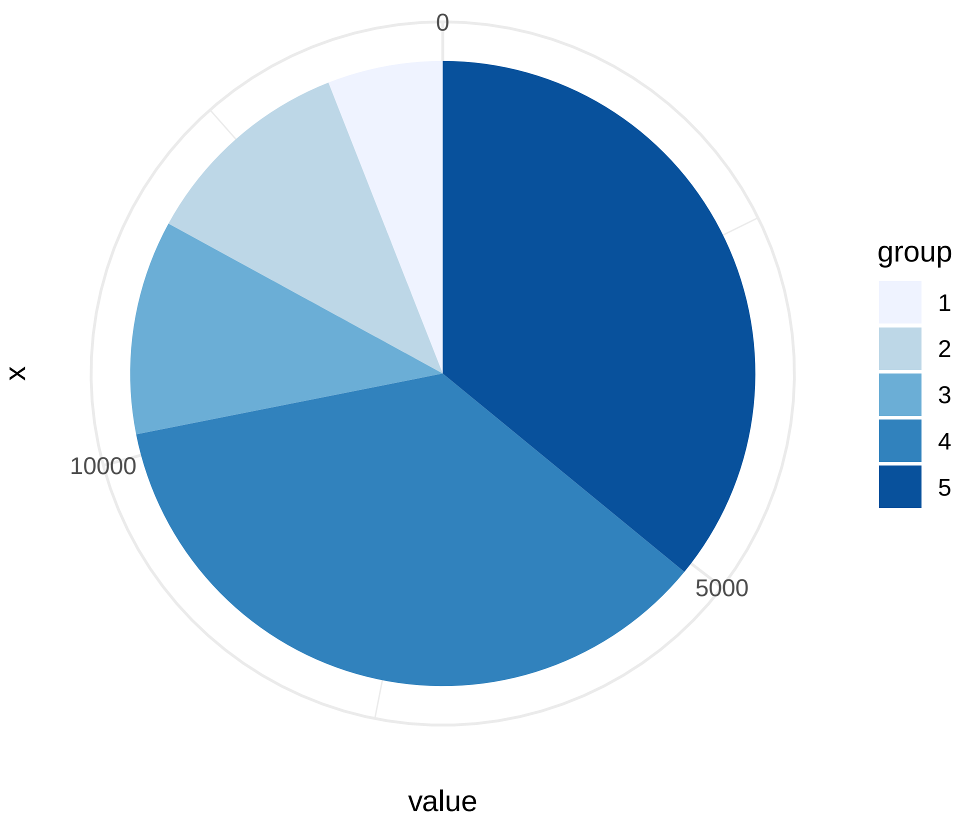
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



As we can see, all values seem to correspond to their intended meanings, with no outliers or invalid values (for example a Rating of -1). Also, with the histogram we are starting to visualise and explore the data, wich comes next.

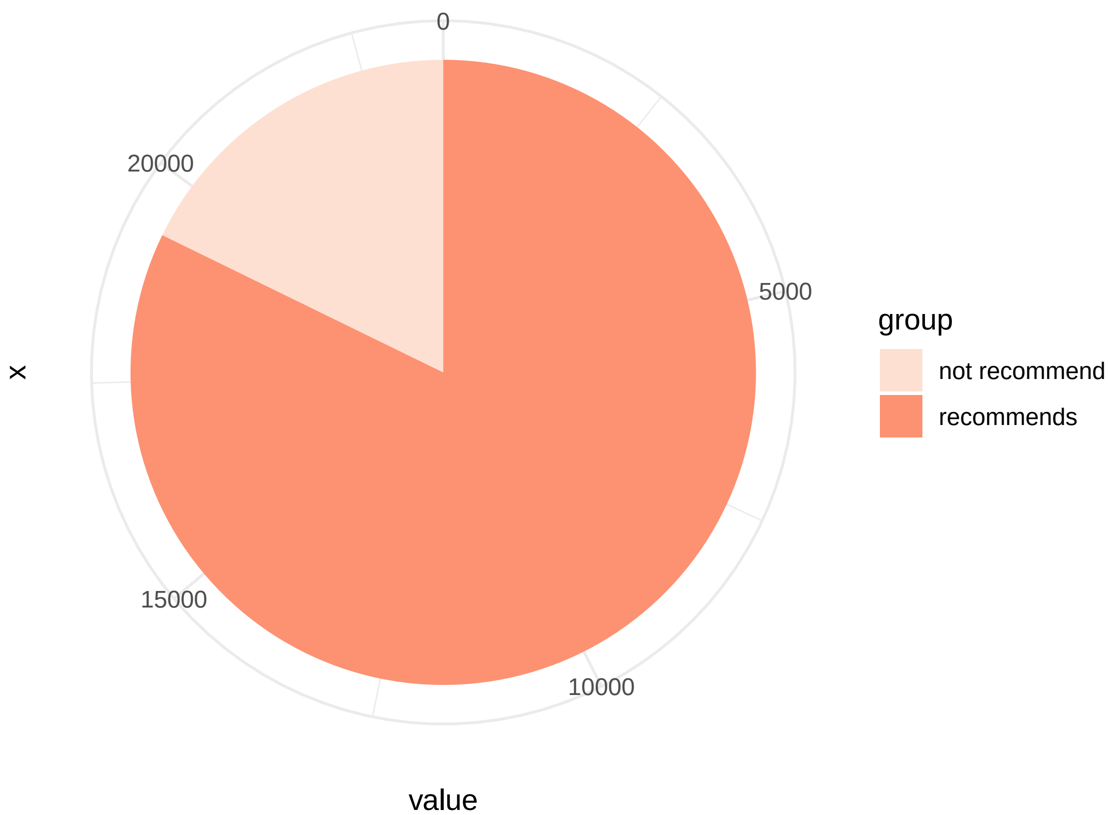
Exploratory analisis

```
df <- data.frame(
  group = c("1", "2", "3", "4", "5"),
  value = c(sum(data$Rating==1), sum(data$Rating==2), sum(data$Rating==2),sum(data$Rating==4),sum(data$R
  )
 ggplot(df, aes(x="", y=value, fill=group)) + geom_bar(width = 1, stat = "identity")+ coord_polar("y", s
  theme_minimal()
```



We can see that most of the customers liked their products and are satisfied. Almost 2 thirds of the reviews have a rating of 4 or bigger. This can hit at the fact that most customers would recommend the product purchased.

```
df <- data.frame(
  group = c("recommends", "not recommend"),
  value = c(sum(data$Recommend==1), sum(data$Recommend==0))
  )
 ggplot(df, aes(x="", y=value, fill=group)) + geom_bar(width = 1, stat = "identity")+ coord_polar("y", s
  theme_minimal()
```



Confirming our past hypothesis, clearly, more customers recommended the clothes in front of the less than a third of customers that did not. This will be a problem, because we would like balanced data, where we have a 50/50 split between happy and not so happy customers.

Text preprocessing: normalization, removing non-letter characters, removing stopwords and stemming

Now that we have a general idea of the distribution of the reviews let's normalize, remove "wierd" characters, and eliminate those words that bring no useful information (stopwords)

```
summary(data$corpus <- corpus(data$all_text),10)
```

```
## Corpus consisting of 23486 documents, showing 10 documents:
##
##    Text Types Tokens Sentences
##   text1     7      8         1
##   text2    50     70         2
##   text3    77    115         2
##   text4    25     34         3
##   text5    30     43         1
##   text6    67    112         1
##   text7    82    124         1
##   text8    79    123         1
##   text9    34     38         1
```

```
## text10    67     93        2
```

```r
data$tokens = tokens(data$corpus, remove_numbers = TRUE, remove_punct = TRUE, remove_separators = TRUE)
#data$dfm = dfm(data$tokens, remove = stopwords("english"), tolower=TRUE, stem = TRUE)
data$dfm = dfm(data$tokens, tolower=TRUE) %>% dfm_remove( stopwords_en) %>% dfm_wordstem()
summary(data$dfm)
```

```
##    Length     Class      Mode
## 299117696       dfm        S4
```

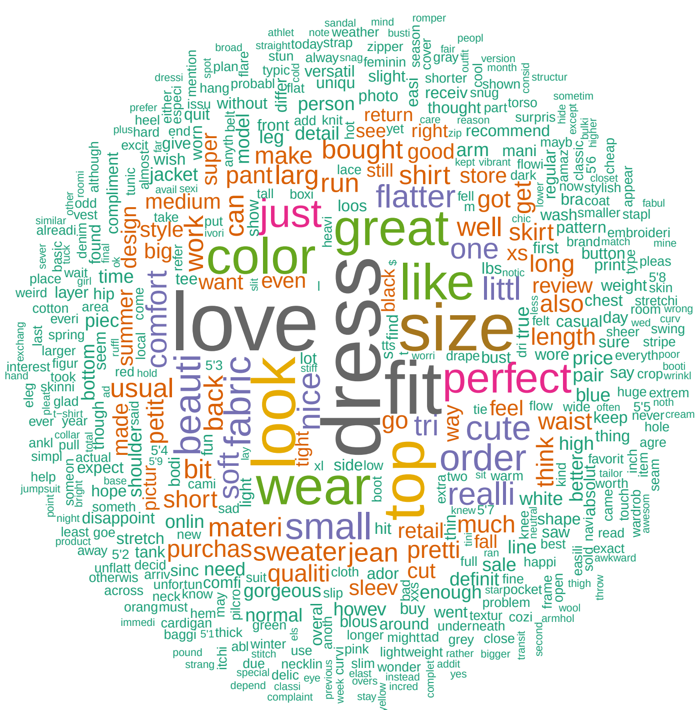
With our Quanteda functions we can convert the text into a series of tokens, and in turn, those tokens into a Document-Feature Matrix, which will be the data used by the models. Our DFM, contains all words, with al characters converted to lowercase, without stopwords and stemmed. Stemming is the process of converting a word into it's base form, facilitating the model learning process.

Let's see graphically our DFM information, like which words are more common.

```r
topfeatures(data$dfm, 20) # 20 most frequent words
```

```
##   dress    love     fit    size    look     top    wear    like   great   color
##   13812   13808   12100   10946    9576    9503    8201    7978    7901    7745
## perfect    just  fabric   order   small  beauti    cute    nice flatter  realli
##    6163    5828    5170    5027    4981    4963    4589    4355    4174    4073
```

```r
textplot_wordcloud(data$dfm, random_order = FALSE,
                   rotation = .25,
                   color = RColorBrewer::brewer.pal(8, "Dark2"))
```



Intuitively, words like “dress”, “top” or “size” are very common due to the nature of our data. Other words like “love” or “perfect” indicate, again, that most of those reviews are positive; menwhile, words like “return”, indicating a desire to return the product, are much less common.

Model training

First, we will be spliting our data into train and test set. About 80% of the reviews will be used for training.

```
train_index <- createDataPartition(data$id, p = .8, list = FALSE, times = 1)
train <- subset(data, data$id %in% train_index)
`%notin%` <- Negate(`%in%`)
test <- subset(data, data$id %notin% train_index)
```

We will be using the Naive Bayes model provided by Quanteda. One of the parameters of that model is prior. So we will test all 3 possible options for that parameter to get the best possible result.

```
#Naive Bayes text model with prior distribution on text set to 'termfreq'
nb_model <- textmodel_nb(train$dfm, train$Recommend, smooth=1, prior="termfreq")
summary(nb_model)
```

```
##
## Call:
## textmodel_nb.dfm(x = train$dfm, y = train$Recommend, smooth = 1,
##     prior = "termfreq")
##
## Class Priors:
## (showing first 2 elements)
##      0      1
## 0.1952 0.8048
##
## Estimated Feature Scores:
##     absolut    wonder     silki      sexi  comfort    love   dress      sooo
## 0 0.0007845 0.0004319 0.0001410 9.697e-05 0.001402 0.01163 0.01582 0.0001322
## 1 0.0013791 0.0008638 0.0001989 4.383e-04 0.006173 0.02083 0.01987 0.0000727
##     pretti    happen      find    store      glad        bc     never    order
## 0 0.003658 0.0003879 0.0009344 0.002133 0.0001939 0.0001587 0.0006082 0.007131
## 1 0.003986 0.0001796 0.0016699 0.003036 0.0010221 0.0001005 0.0006222 0.006951
##      onlin    petit   bought       5’8   length       me-      hit     just
## 0 0.002750 0.002618 0.002177 0.0005113 0.002318 3.526e-05 0.000952 0.009300
## 1 0.002155 0.003836 0.004561 0.0004854 0.004037 3.635e-05 0.001345 0.007715
##      littl      knee  definit     true      midi    someon
## 0 0.002653 0.0006611 0.001331 0.001093 4.408e-05 0.0010843
## 1 0.006049 0.0009943 0.002185 0.002703 1.219e-04 0.0004341
```

```
prediction = predict(nb_model, newdata = test$dfm)
cat("Accuaracy of the termfreq model: ",(sum(prediction == test$Recommend)/count(test))$n)
```

```
## Accuaracy of the termfreq model:  0.8918459
```

```r
#Naive Bayes text model with prior distribution on text set to 'uniform'
nb_model_uniform <- textmodel_nb(train$dfm, train$Recommend, smooth=1, prior="uniform")

prediction = predict(nb_model_uniform, newdata = test$dfm)
cat("Accuaracy of the termfreq model: ",(sum(prediction == test$Recommend)/count(test))$n)
```

```
## Accuaracy of the termfreq model:  0.8520332
```

```r
#Naive Bayes text model with prior distribution on text set to 'docfreq'
nb_model_docfreq <- textmodel_nb(train$dfm, train$Recommend, smooth=1, prior="docfreq")

prediction = predict(nb_model_docfreq, newdata = test$dfm)
cat("Accuaracy of the termfreq model: ",(sum(prediction == test$Recommend)/count(test))$n)
```

```
## Accuaracy of the termfreq model:  0.8920588
```

As we see, according to the accuaracy measure, the "docfreq" option allows for a slight advantage in front of the others.

Now, as we said earlier, this dataset is not balanced. So to corret this, we will make a subset of the data that includes all negative reviews and the same amount of positive reviews, in order to get a perfect 50% split.

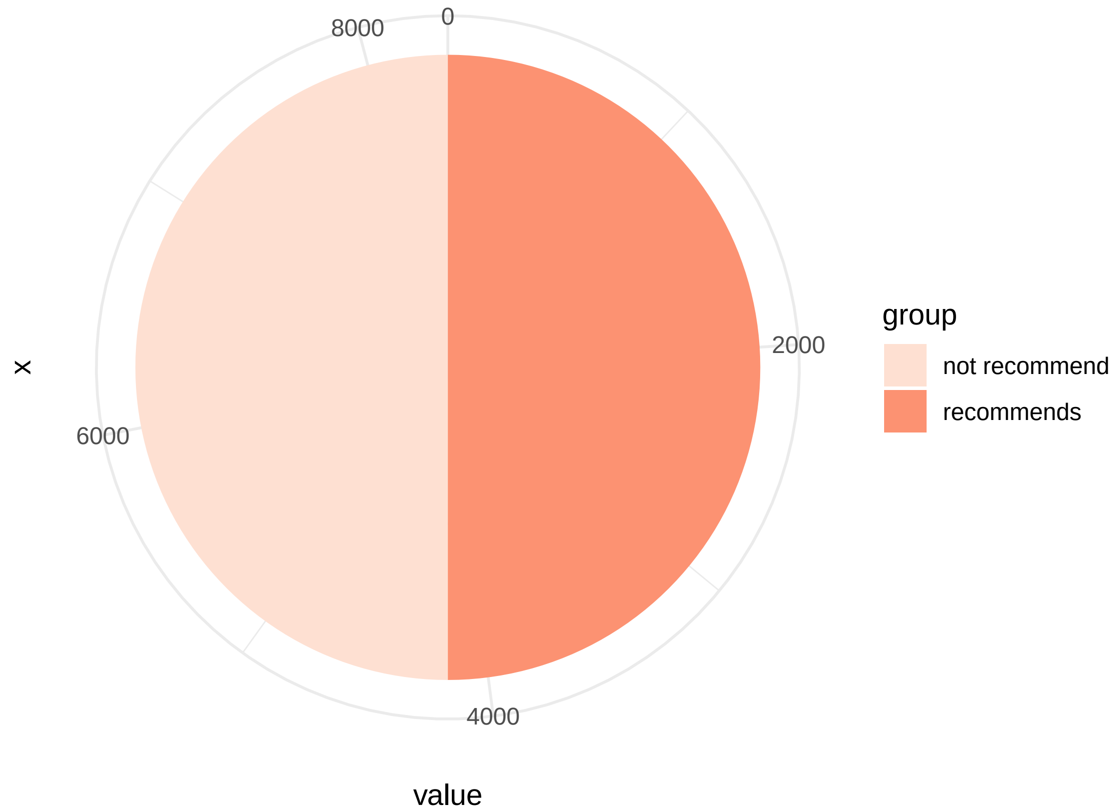
```r
#separate positivo from negative
data_reduced <- subset(data, select=-c(corpus,tokens,dfm))
pos <- data_reduced[data_reduced$Recommend==1,]
neg <- data_reduced[data_reduced$Recommend==0,]

i = as.integer(count(neg)$n[1])
reduced_pos <- pos[sample(1:nrow(pos), size = i), ]

balanced_data = bind_rows(reduced_pos, neg)
```

To create the balanced_data we needed to remove de corpus, tokens and dfm columns. Let's create them back.

```r
df <- data.frame(
  group = c("recommends", "not recommend"),
  value = c(sum(balanced_data$Recommend==1), sum(data$Recommend==0))
  )
 ggplot(df, aes(x="", y=value, fill=group)) + geom_bar(width = 1, stat = "identity") + coord_polar("y",
```



As we see, we got that split, where half of the reviews are positive, and half are negative. Lets create again the train and test set and test some models.

```
balanced_data$corpus <- corpus(balanced_data$all_text)
balanced_data$tokens = tokens(balanced_data$corpus, remove_numbers = TRUE, remove_punct = TRUE, remove_
balanced_data$dfm = dfm(balanced_data$tokens, tolower=TRUE) %>% dfm_remove( stopwords_en) %>% dfm_words

train_index_b <- createDataPartition(balanced_data$id, p = .8, list = FALSE, times = 1)
train_b <- subset(balanced_data, balanced_data$id %in% train_index)
test_b <- subset(balanced_data, balanced_data$id %notin% train_index)
```

```
#Naive Bayes text model with prior distribution on text set to 'docfreq'
nb_model_b <- textmodel_nb(train_b$dfm, train_b$Recommend, smooth=1, prior="docfreq")
summary(nb_model_b)
```

```
##
## Call:
## textmodel_nb.dfm(x = train_b$dfm, y = train_b$Recommend, smooth = 1,
##     prior = "docfreq")
##
## Class Priors:
## (showing first 2 elements)
##      0      1
## 0.4975 0.5025
##
## Estimated Feature Scores:
```

```
##       cute      cozi  sweater grand-dad    style    love  pattern   fabric
## 0 0.005472 0.0002487 0.003961 9.212e-06 0.002662 0.01215 0.001713 0.010041
## 1 0.006331 0.0007961 0.004158 1.873e-05 0.002960 0.01976 0.001761 0.006219
##        run      bit     larg    especi   length    sleev  definit     fit
## 0 0.003547 0.002238 0.004578 0.0007830 0.002423 0.003003 0.001391 0.01432
## 1 0.004636 0.003943 0.004046 0.0006369 0.003596 0.002613 0.002098 0.01668
##       overs      sort    look      find  comfort  perfect    casual   weekend
## 0 0.0004882 0.0003869 0.01839 0.0009765 0.001465 0.002183 0.0004145 6.448e-05
## 1 0.0004495 0.0002435 0.01125 0.0015266 0.005647 0.009497 0.0020042 3.746e-04
##      outfit    great    short     high    waist      wide
## 0 0.0001290 0.005002 0.004256 0.002110 0.003841 0.0020635
## 1 0.0004495 0.011707 0.003456 0.001986 0.003512 0.0007586
```

```r
prediction = predict(nb_model_b, newdata = test_b$dfm)
cat("Accuaracy of the termfreq model: ",(sum(prediction == test$Recommend)/count(test))$n)
```

```
## Warning in `==.default`(prediction, test$Recommend): longer object length is not
## a multiple of shorter object length
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
## Accuaracy of the termfreq model:  0.4937194
```

The accuracy decreased. This is due to the fact that now, that the data is balanced, it's quite harder to predict correctly values. However, this allows for a more "reliable" model, less "overfitted" to our unbalanced data, and more fit to work in the real world.

Let's use the confusion matrix to evaluate the model obtained.

```r
confusionMatrix(prediction, factor(test_b$Recommend))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 732 121
##          1 117 694
##
##                Accuracy : 0.857
##                  95% CI : (0.8392, 0.8735)
##     No Information Rate : 0.5102
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7138
##
##  Mcnemar's Test P-Value : 0.8458
##
##             Sensitivity : 0.8622
##             Specificity : 0.8515
##          Pos Pred Value : 0.8581
##          Neg Pred Value : 0.8557
```

```
##               Prevalence : 0.5102
##           Detection Rate : 0.4399
##     Detection Prevalence : 0.5126
##        Balanced Accuracy : 0.8569
##
##         'Positive' Class : 0
##
```

```r
precision <- 679/(679+97)
recall <- 679/(679+138)
cat("precision: ",precision,", recall: ",recall)
```

```
## precision:  0.875 , recall:  0.8310894
```

As we see, our model still holds a acc of 0.857, which is still quite high. Also, the precision and recall measures, together with the confusion matrix indicate that the model is quite balanced, meaning that it doesn't fail more at False Positives or at False Negatives, instead, it fails, more or less, the same amount on both cases.

References

- https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews
- https://www.marsja.se/how-to-concatenate-two-columns-or-more-in-r-stringr-tidyr/
- https://discuss.analyticsvidhya.com/t/how-to-count-the-missing-value-in-r/2949/4
- https://www.r-graph-gallery.com/index.html
- https://www.r-bloggers.com/2021/05/sentiment-analysis-in-r-3/
- https://cran.r-project.org/web/packages/corpus/vignettes/stemmer.html
- https://quanteda.io/articles/quickstart.html#extracting-features-from-a-corpus-1
- https://www.journaldev.com/46732/confusion-matrix-in-r