Assignment 2

Simon

13/10/2021

```r
### Load standardpackages
library(tidyverse) # Collection of all the good stuff like dplyr, ggplot2 ect.
library(magrittr) # For extra-piping operators (eg. %<>%)
library(textrecipes)
library(recipes)
library(stopwords)
library(tidytext)
library(readr)
library(SnowballC)
library(topicmodels)
```

Loader datasættet

```r
data_start <- read_csv("https://raw.githubusercontent.com/simonmig10/M2-sds/main/twitter_hate_speech.cs
```

```
## New names:
## * `` -> ...1

## Rows: 24783 Columns: 3

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): tweet
## dbl (2): ...1, class

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
  #rename(X1 = ...1)

data_start %>% as.tibble()
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.

## # A tibble: 24,783 x 3
##     ...1 class tweet
```

```
##    <dbl> <dbl> <chr>
##  1     0     2 "!!! RT @mayasolovely: As a woman you shouldn't complain about c~
##  2     1     1 "!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat ~
##  3     2     1 "!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever f~
##  4     3     1 "!!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny"
##  5     4     1 "!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me mi~
##  6     5     1 "!!!!!!!!!!!!!!!!!!\"@T_Madison_x: The shit just blows me..claim~
##  7     6     1 "!!!!!!\"@__BrighterDays: I can not just sit up and HATE on anot~
##  8     7     1 "!!!!&#8220;@selfiequeenbri: cause I'm tired of you big bitches ~
##  9     8     1 "\" &amp; you might not get ya bitch back &amp; thats that \""
## 10     9     1 "\" @rhythmixx_ :hobbies include: fighting Mariam\"\n\nbitch"
## # ... with 24,773 more rows
```

1. Preprocessing

Removing retweets and removing numbers from tweets.

```
data_start$tweet = data_start$tweet %>%
  str_remove_all("[0123456789]")

data_start %<>%
  filter(!(tweet %>% str_detect('RT')))  ##%>%
  ##rename(ID = X1)
```

Checking that retweets got removed

```
data_start %>% glimpse()
```

```
## Rows: 17,615
## Columns: 3
## $ ...1  <dbl> 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2~
## $ class <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ tweet <chr> "!!!!!!!!!!!!!!!!!!\"@T_Madison_x: The shit just blows me..claim~
```

Converting data to a tibble so it can be tokenized

```
data <- tibble(ID = data_start[[1]] %>% as.numeric(),
                text = data_start[[3]] %>% as.character(),
                labels = data_start[[2]] %>% as.logical())

data_sml= tibble(ID = data_start[[1]] %>% as.numeric(),
                text = data_start[[3]] %>% as.character(),
                labels = data_start[[2]])

data_sml_2 = data_sml %>%
  filter(labels %in% c(0,1))%>%
  mutate(labels= ifelse(labels== 0, "Hate", "offens"))
```

tokenizing the data by tweets and stemming the words, so singular and plural words become the same.

```
data_tidy <- data %>%
  unnest_tokens(word, text, token = "tweets")
```

```
## Using `to_lower = TRUE` with `token = 'tweets'` may not preserve URLs.
```

```
  #text_tokens(word, stemmer = "en") %>%
```

Checking whether it worked

```
data_tidy %>% head(50)
```

```
## # A tibble: 50 x 3
##       ID labels word
##    <dbl> <lgl>  <chr>
##  1     5 TRUE   tmadisonx
##  2     5 TRUE   the
##  3     5 TRUE   shit
##  4     5 TRUE   just
##  5     5 TRUE   blows
##  6     5 TRUE   meclaim
##  7     5 TRUE   you
##  8     5 TRUE   so
##  9     5 TRUE   faithful
## 10     5 TRUE   and
## # ... with 40 more rows
```

It did, so now we count the words to check the most used words

```
data_tidy %>% count(word, sort = TRUE)
```

```
## # A tibble: 26,333 x 2
##    word      n
##    <chr> <int>
##  1 a      6352
##  2 bitch  5932
##  3 i      5499
##  4 the    5004
##  5 you    4100
##  6 to     3633
##  7 and    2803
##  8 my     2645
##  9 that   2536
## 10 in     2103
## # ... with 26,323 more rows
```

It shows that irrelevant words like “i” and “a” are commonly present, so these needs to be removed.

Hashtags are being removed and so are other typical twitter specific stuff like http and so on. Words shorter than 3 letters are also removed and words occuring less than 100 times. Lastly the data is antijoined with stopwords and the tidying proces is done.

```r
# preprocessing
data_tidy %<>%
  filter(!(word %>% str_detect('@'))) %>% # remove hashtags and mentions
  filter(!(word %>% str_detect('^amp|^http|^t\\.co'))) %>% # Twitter specific stuff
#  mutate(word = word %>% str_remove_all('[^[:alnum:]]')) %>% ## remove all special characters
  filter(str_length(word) > 2 ) %>% # Remove words with less than  3 characters
  group_by(word) %>%
  filter(n() > 100) %>% # remove words occuring less than 100 times
  ungroup() %>%
  anti_join(stop_words, by = 'word') %>%  # remove stopwords
  mutate(word = wordStem(word))
```

TF IDF'

The TF_IDF score of each word is now being added to the tidied data. The TF-IDF (term frequency-inverse document frequency) is a measure that evaluates how relevant a word is to a document in a collection of documents.

This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

```r
# TFIDF weights
data_tidy %<>%
  add_count(ID, word) %>%
  bind_tf_idf(term = word,
              document = ID,
              n = n)
```

Now the scores are showed.

```r
# TFIDF topwords
data_tidy %>%
  count(word, wt = tf_idf, sort = TRUE) %>%
  head(10)
```

```
## # A tibble: 10 x 2
##    word      n
##    <chr> <dbl>
##  1 bitch 3012.
##  2 hoe   2595.
##  3 pussi 1950.
##  4 trash 1449.
##  5 fuck  1296.
##  6 dont  1062.
##  7 ass   1005.
##  8 nigga  990.
##  9 bird   985.
## 10 lol    932.
```

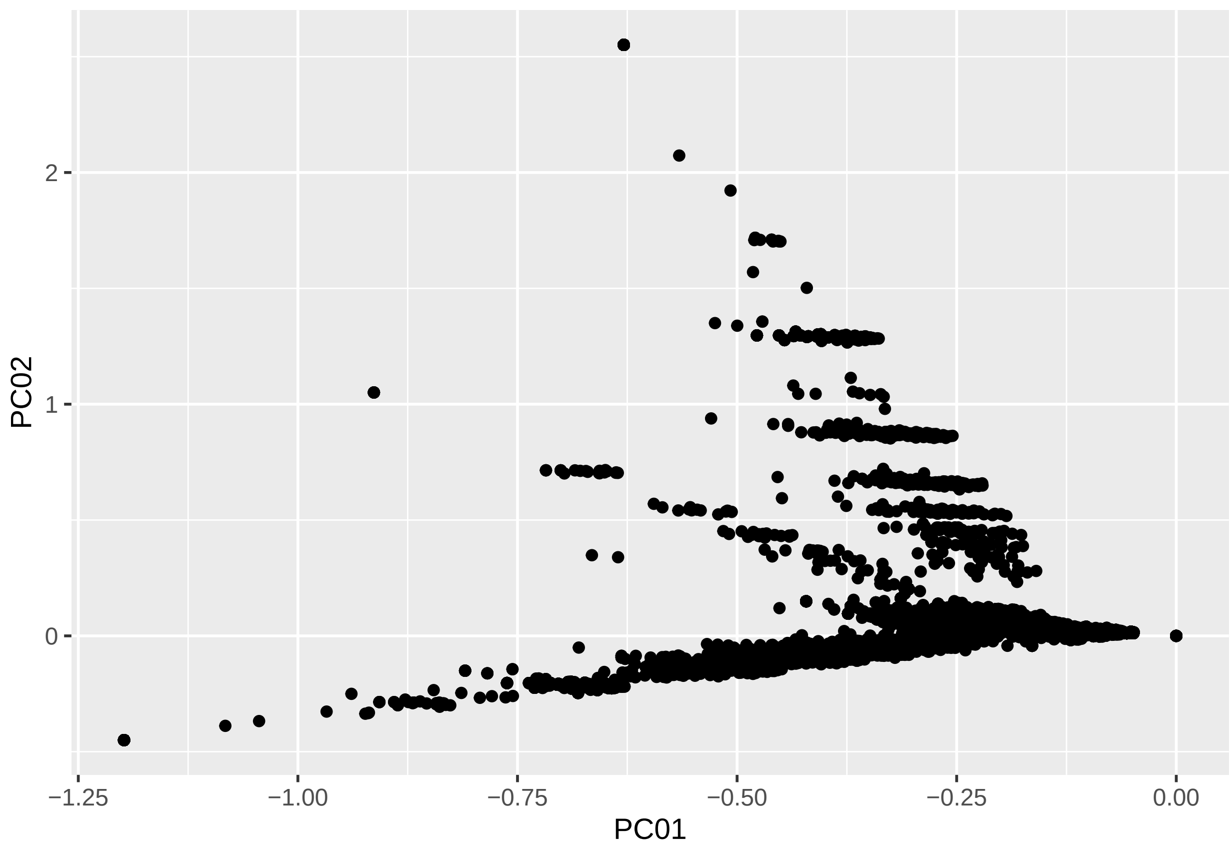
And here the ten words with the highest TF_IDF displayed.

Dimensionality reduction

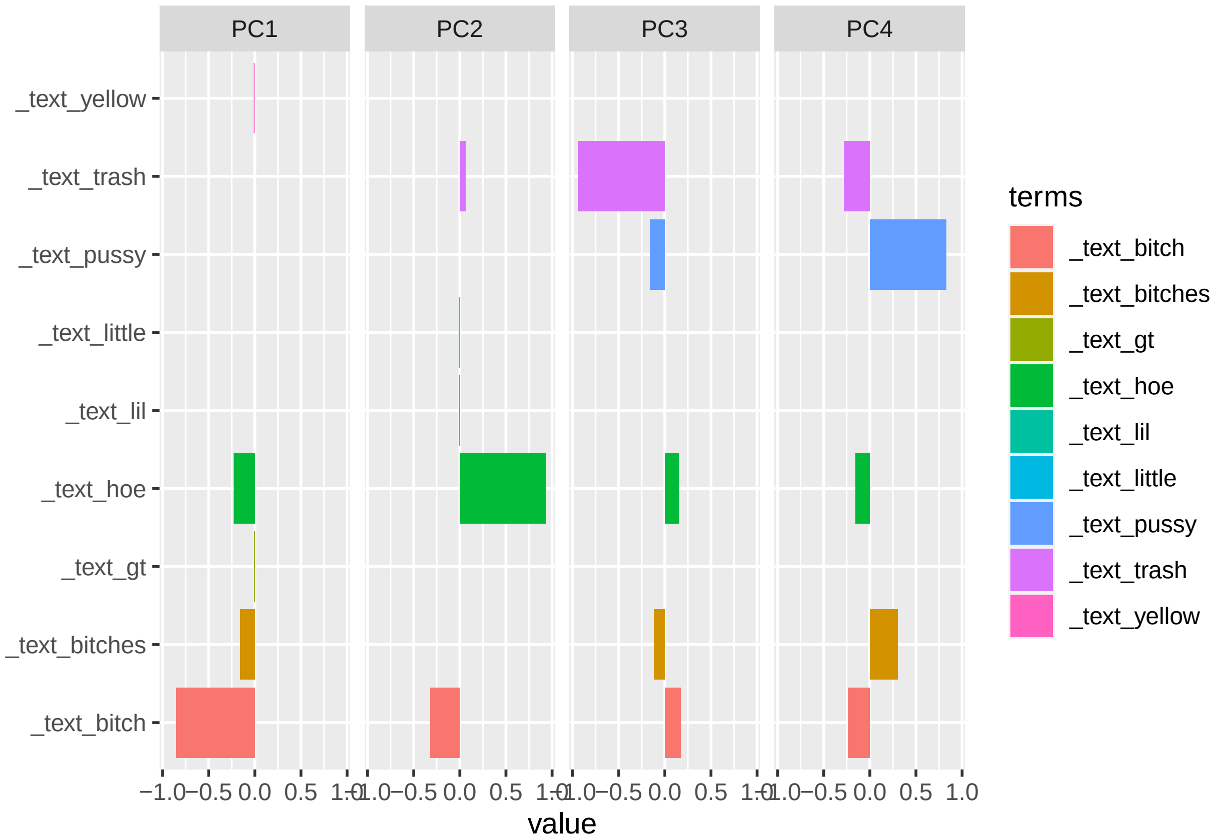
Creating a recipe

A recipe is created to dimensionality reduce the data and weighting them by TD-IDF scores.

```
recipe_base <- data %>%
  select(ID, text) %>%
  # BAse recipe starts
  recipe(~.) %>%
  update_role(ID, new_role = "ID") %>% # Update role of ID
  step_tokenize(text, token = 'words') %>% # tokenize
  step_stopwords(text, keep = FALSE) %>% # remove stopwords
  step_untokenize(text) %>% # Here we now have to first untokenize
  step_tokenize(text, token = "ngrams", options = list(n = 1, n_min = 1)) %>% # and tokenize again
  step_tokenfilter(text, min_times = 25) %>%
  prep()

recipe_pca <- recipe_base %>% # tokenize
  step_tfidf(text, prefix = '') %>% # TFIDF weighting --> so different from the above.
  step_pca(all_predictors(), num_comp = 10) %>% # PCA
  prep()

#Plot 1
recipe_pca %>% juice() %>%
  ggplot(aes(x = PC01, y = PC02)) +
  geom_point()
```




```r
#Plot 2
recipe_pca %>%
  tidy(7) %>%
  filter(component %in% paste0("PC", 1:4)) %>%
  group_by(component) %>%
    arrange(desc(value)) %>%
    slice(c(1:2, (n()-2):n())) %>%
  ungroup() %>%
  mutate(component = fct_inorder(component)) %>%
  ggplot(aes(value, terms, fill = terms)) +
  geom_col(show.legend = TRUE) +
  facet_wrap(~component, nrow = 1) +
  labs(y = NULL)
```



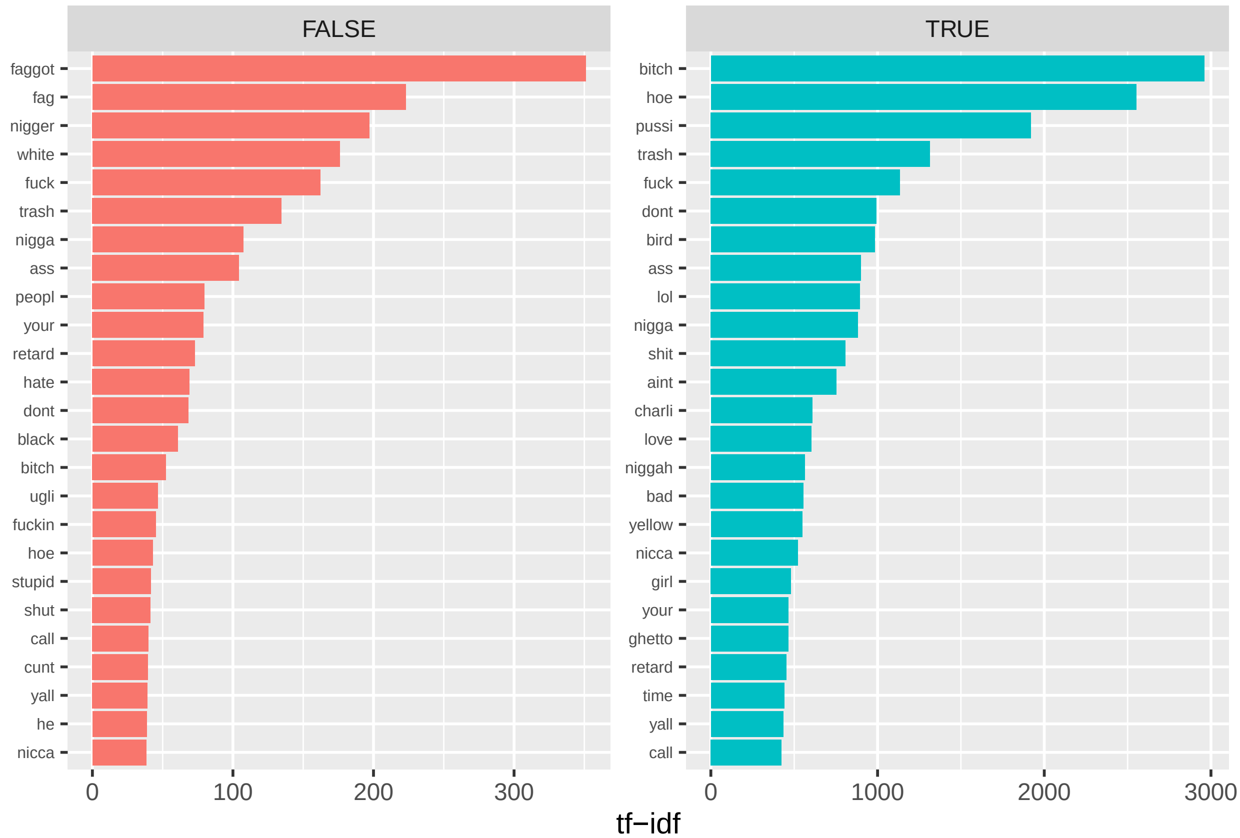
These plots are not just awesome they also show how well some of the highest scored words are explained by each principal component.

2. Explore and compare the 2 "classes of interest" - hate speech vs offensive language

Can you see differences by using simple count-based approaches?

```
labels_words <- data_tidy %>%
  group_by(labels) %>%
  count(word, wt = tf_idf, sort = TRUE, name = "tf_idf") %>%
  slice(1:25) %>%
  ungroup()
```

```
labels_words %>%
  mutate(word = reorder_within(word, by = tf_idf, within = labels)) %>%
  ggplot(aes(x = word, y = tf_idf, fill = labels)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~labels, ncol = 2, scales = "free") +
  coord_flip() +
  scale_x_reordered() +
  theme(axis.text.y = element_text(size = 6))
```



The result shows that the words with the highest TD-IDF score in the hate speech basket is “faggot”, “fag” and “nigger” where as the words with the highest TD-IDF in the offensive language basket is “bitch”, “hoe” and “pussi”. This shows that hate speech is associated with racism and homophobia, where as offensive language is more cussing and bad mouthing.

##Can you identify themes (aka clusters / topics) that are specific for one class or another?

LDA-topic modelling is used to separate the dataset into topics for hate speech and offensive language respectively.

First the tidy data is extracted and split into two datasets

```
lda_data = data_tidy %>%
  select(ID, labels, word, n, tf_idf)

off = lda_data %>%
  filter(labels == TRUE) %>%
  as.tibble()

hate = lda_data %>%
  filter(labels == FALSE) %>%
  as.tibble()
```

Then the data is made into a document-term matrix

```
text_dtm1 <- hate %>%
  cast_dtm(document = ID, term = word, value = n)
```

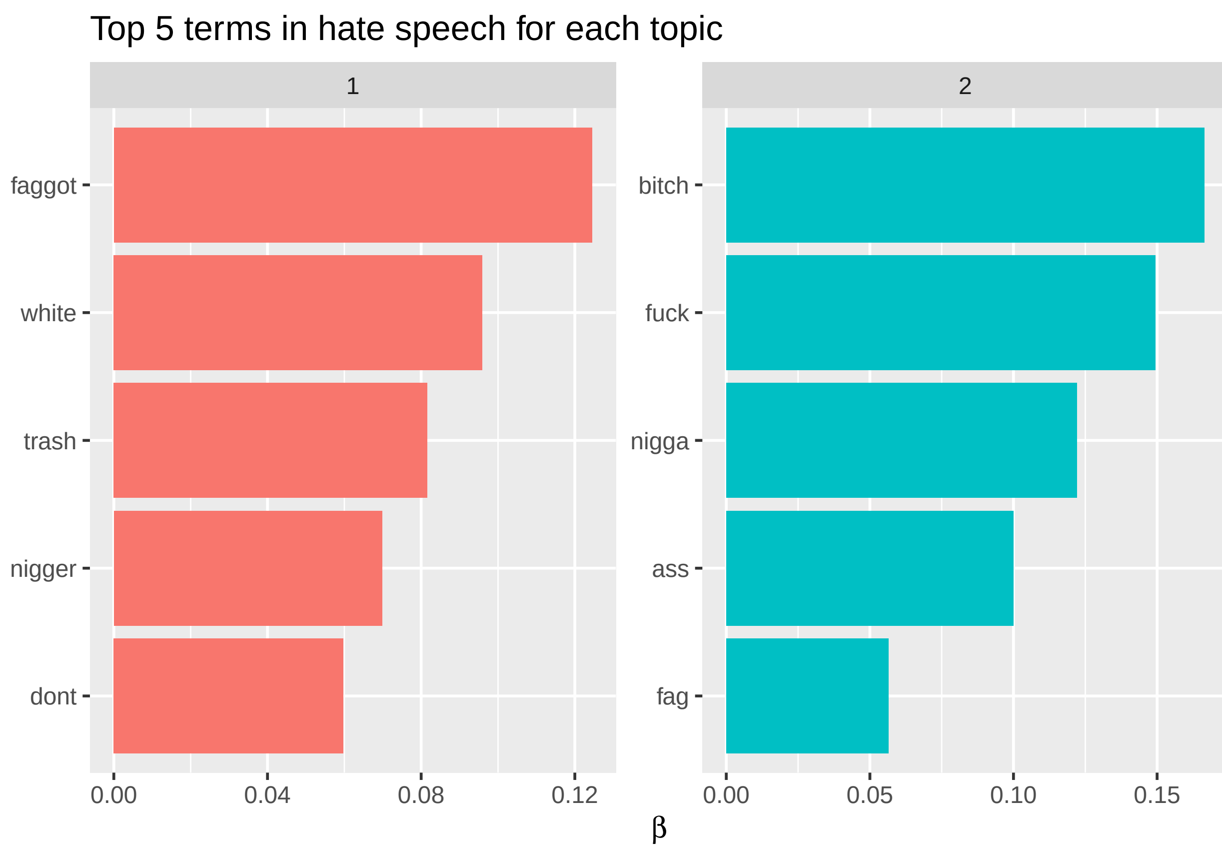
```r
text_dtm2 <- off %>%
  cast_dtm(document = ID, term = word, value = n)
```

The topics are created using the "Gibbs" method and there are only being created two topics for each category

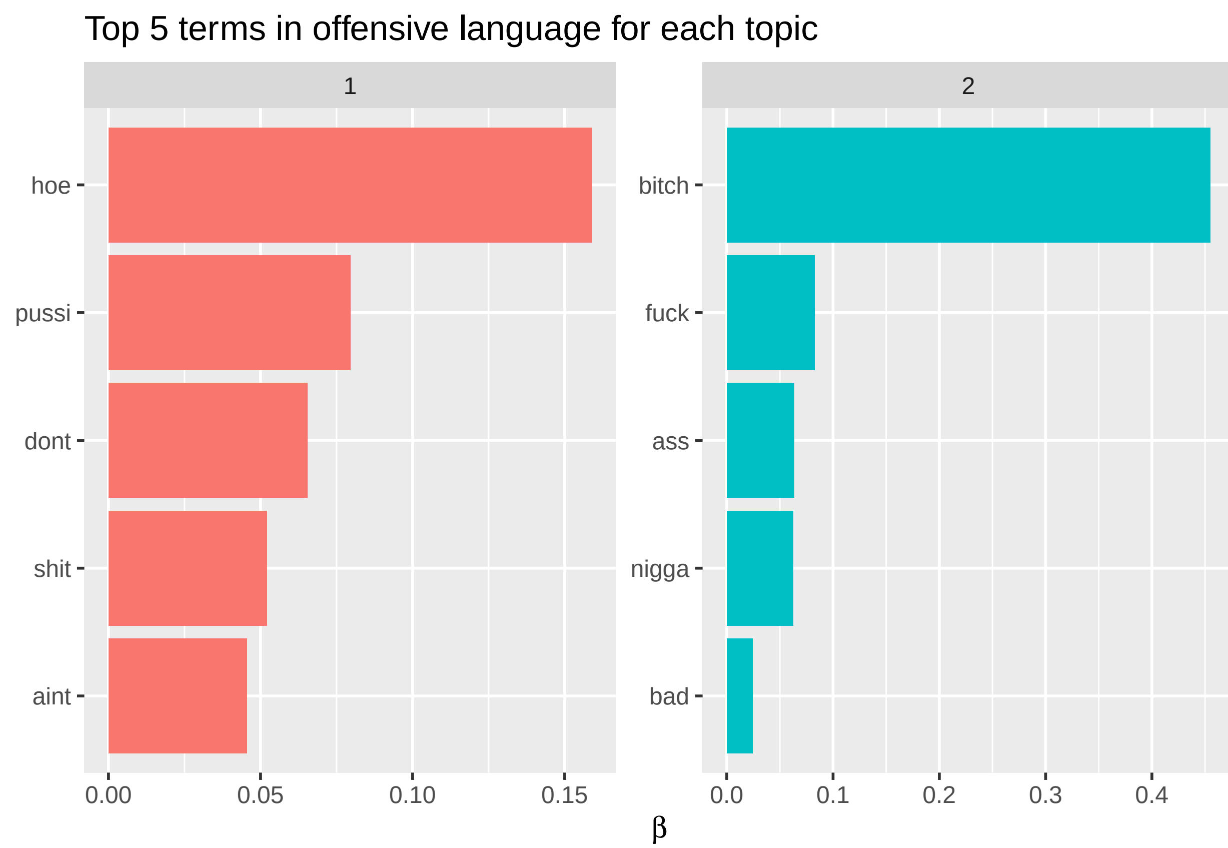
```r
text_lda1 <- text_dtm1 %>%
  LDA(k = 2, method = "Gibbs",
      control = list(seed = 1337))

text_lda2 <- text_dtm2 %>%
  LDA(k = 2, method = "Gibbs",
      control = list(seed = 1337))
```

We want the Beta parameter as i shows the probability that a word occurs in a certain topic. Therefore, looking at the top probability words of a topic often gives us a good intuition regarding its properties, which is done in two plots below.

```r
lda_beta1 <- text_lda1 %>%
  tidy(matrix = "beta")

lda_beta2 <- text_lda2 %>%
  tidy(matrix = "beta")

lda_beta1 %>%
  # slice
  group_by(topic) %>%
  arrange(topic, desc(beta)) %>%
  slice(1:5) %>%
  ungroup() %>%
  # visualize
  mutate(term = reorder_within(term, beta, topic)) %>%
  group_by(topic, term) %>%
  arrange(desc(beta)) %>%
  ungroup() %>%
  ggplot(aes(term, beta, fill = as.factor(topic))) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  scale_x_reordered() +
  labs(title = "Top 5 terms in hate speech for each topic",
       x = NULL, y = expression(beta)) +
  facet_wrap(~ topic, ncol = 3, scales = "free")
```




```
lda_beta2 %>%
  # slice
  group_by(topic) %>%
  arrange(topic, desc(beta)) %>%
  slice(1:5) %>%
  ungroup() %>%
  # visualize
  mutate(term = reorder_within(term, beta, topic)) %>%
  group_by(topic, term) %>%
  arrange(desc(beta)) %>%
  ungroup() %>%
  ggplot(aes(term, beta, fill = as.factor(topic))) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  scale_x_reordered() +
  labs(title = "Top 5 terms in offensive language for each topic",
       x = NULL, y = expression(beta)) +
  facet_wrap(~ topic, ncol = 3, scales = "free")
```



The two above plots kinda shows a similar picture as the one showcased earlier. It seems like the two topics are not separated by any particular measure such as racism and homophobia in the case of hate speech nor seems the topic in offensive language to be separated by anything specific.

3. Build an ML model that can predict hate speech

We want to use the tidymodels package to build the models.

```
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip

## -- Attaching packages -------------------------------------- tidymodels 0.1.3 --

## v broom        0.7.9      v rsample      0.1.0
## v dials        0.0.10     v tune         0.1.6
## v infer        1.0.0      v workflows    0.2.3
## v modeldata    0.1.1      v workflowsets 0.1.0
## v parsnip      0.1.7      v yardstick    0.0.8

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard()     masks purrr::discard()
```

```
## x magrittr::extract()   masks tidyr::extract()
## x dplyr::filter()       masks stats::filter()
## x recipes::fixed()      masks stringr::fixed()
## x dplyr::lag()          masks stats::lag()
## x magrittr::set_names() masks purrr::set_names()
## x yardstick::spec()     masks readr::spec()
## x recipes::step()       masks stats::step()
## * Use tidymodels_prefer() to resolve common conflicts.
```

Simple manual baseline

We create a mean model.

```
words_classifier <- labels_words %>%
  arrange(desc(tf_idf)) %>%
  distinct(word, .keep_all = TRUE) %>%
  select(-tf_idf)
```

```
tweet_null_model <- data_tidy %>%
  inner_join(labels_words, by = 'word')
```

```
null_res <- tweet_null_model %>%
  group_by(ID) %>%
  summarise(truth = mean(labels.x, na.rm = TRUE) %>% round(0),
            pred = mean(labels.y, na.rm = TRUE) %>% round(0))
```

```
table(null_res$truth, null_res$pred)
```

```
##
##         0    1
##   0   782  124
##   1  9426 4943
```

We can see the model is very bad at predicting hate speech, with 0 beeing hate speech and 1 beeing offensive language.

Training & Test split

We create a training and test dataset

```
data_sml_2 %<>%
  #filter(!(text %>% str_detect('@'))) %>% # remove hashtags and mentions
  #filter(!(text %>% str_detect('^amp|^http|^t\\.co'))) %>% # Twitter specific stuff
  select(-ID) %>%
  rename(y = labels) %>%
  mutate(y= y %>% as.factor())
```

```
data_split_test= initial_split(data_sml_2, prop = 0.75, strata = y)



data_split <- initial_split(data_sml_2, prop = 0.75, strata = y)

data_train <- data_split  %>%  training()
data_test <- data_split %>% testing()
```

Preprocessing pipeline

We create a recipe for a model with and without dimensionality reduction.

```
# This recipe pretty much reconstructs all preprocessing we did so far
data_recipe <- data_train %>%
  recipe(y ~.) %>%
  themis::step_downsample(y) %>% # For downsampling class imbalances (optimal)
  step_filter(!(text %>% str_detect('^RT'))) %>% # Upfront filtering retweets
  step_filter(text != "@") %>%
  step_tokenize(text, token = "tweets") %>% # tokenize
  step_tokenfilter(text, min_times = 75) %>%  # Filter out rare words
  step_stopwords(text, keep = FALSE) %>% # Filter stopwords
  step_tfidf(text) %>% # TFIDF weighting
  #step_pca(all_predictors()) %>% # Dimensionality reduction via PCA (optional)
  prep() # NOTE: Only prep the recipe when not using in a workflow
```

```
## Registered S3 methods overwritten by 'themis':
##   method                  from
##   bake.step_downsample    recipes
##   bake.step_upsample      recipes
##   prep.step_downsample    recipes
##   prep.step_upsample      recipes
##   tidy.step_downsample    recipes
##   tidy.step_upsample      recipes
##   tunable.step_downsample recipes
##   tunable.step_upsample   recipes
```

```
# This recipe pretty much reconstructs all preprocessing we did so far
data_recipe_pca <- data_train %>%
  recipe(y ~.) %>%
  themis::step_downsample(y) %>% # For downsampling class imbalances (optimal)
  step_filter(!(text %>% str_detect('^RT'))) %>% # Upfront filtering retweets
  step_filter(text != "") %>%
  step_tokenize(text, token = "tweets") %>% # tokenize
  step_tokenfilter(text, min_times = 75) %>%  # Filter out rare words
  step_stopwords(text, keep = FALSE) %>% # Filter stopwords
  step_tfidf(text) %>% # TFIDF weighting
  step_pca(all_predictors()) %>% # Dimensionality reduction via PCA (optional)
  prep() # NOTE: Only prep the recipe when not using in a workflow
```

We run the recipies on the trainging and test data.

```
data_train_prep <- data_recipe %>% juice()
data_test_prep <- data_recipe %>% bake(data_test)

data_train_prep_pca <- data_recipe_pca %>% juice()
data_test_prep_pca <- data_recipe_pca %>% bake(data_test)
```

Defining the models

We create the models we want to test including an Elastic net model, random forest model and K-nearest neighbor model

Elastic net model

```
model_en <- logistic_reg(mode = 'classification',
                         mixture = 0.5,
                         penalty = 0.5) %>%
  set_engine('glm', family = binomial)
```

Random forrest model

```
model_rf <-
  rand_forest() %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification")
```

K-nearest neighbor model

```
model_knn <-
  nearest_neighbor(neighbors = 4) %>%
  set_engine("kknn") %>%
  set_mode("classification")
```

fit the model

We fit the models using both the training data for pca and not pca.

```
fit_en <- model_en %>% fit(formula = y ~., data = data_train_prep)

fit_knn <- model_knn %>% fit(formula = y ~., data = data_train_prep)

fit_rf <- model_rf %>% fit(formula = y ~., data = data_train_prep)


##PCA
fit_en_pca <- model_en %>% fit(formula = y ~., data = data_train_prep_pca)
```

```
fit_knn_pca <- model_knn %>% fit(formula = y ~., data = data_train_prep_pca)

fit_rf_pca <- model_rf %>% fit(formula = y ~., data = data_train_prep_pca)
```

We now want to pull the predictions from the fitted models and compare with the true label of each tweet.

```
pred_collected_en <- tibble(
  truth = data_train_prep %>% pull(y),
  pred = fit_en %>% predict(new_data = data_train_prep) %>% pull(.pred_class),
  pred_prob = fit_en %>% predict(new_data = data_train_prep, type = "prob") %>% pull(.pred_offens),
  )

pred_collected_knn <- tibble(
  truth = data_train_prep %>% pull(y),
  pred = fit_knn %>% predict(new_data = data_train_prep) %>% pull(.pred_class),
  pred_prob = fit_knn %>% predict(new_data = data_train_prep, type = "prob") %>% pull(.pred_offens),
  )

pred_collected_rf <- tibble(
  truth = data_train_prep %>% pull(y),
  pred = fit_rf %>% predict(new_data = data_train_prep) %>% pull(.pred_class),
  pred_prob = fit_rf %>% predict(new_data = data_train_prep, type = "prob") %>% pull(.pred_offens),
  )


###PCA
pred_collected_en_pca <- tibble(
  truth = data_train_prep_pca %>% pull(y),
  pred = fit_en_pca %>% predict(new_data = data_train_prep_pca) %>% pull(.pred_class),
  pred_prob = fit_en_pca %>% predict(new_data = data_train_prep_pca, type = "prob") %>% pull(.pred_offen
  )

pred_collected_knn_pca <- tibble(
  truth = data_train_prep_pca %>% pull(y),
  pred = fit_knn_pca %>% predict(new_data = data_train_prep_pca) %>% pull(.pred_class),
  pred_prob = fit_knn_pca %>% predict(new_data = data_train_prep_pca, type = "prob") %>% pull(.pred_off
  )

pred_collected_rf_pca <- tibble(
  truth = data_train_prep_pca %>% pull(y),
  pred = fit_rf_pca %>% predict(new_data = data_train_prep_pca) %>% pull(.pred_class),
  pred_prob = fit_rf_pca %>% predict(new_data = data_train_prep_pca, type = "prob") %>% pull(.pred_offen
  )
```

We can compare them using a confusionmatrix.

```
pred_collected_en %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    699    191
##     offens  170    678
```

```r
pred_collected_rf %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    760    166
##     offens  109    703
```

```r
pred_collected_knn %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    862    599
##     offens    7    270
```

```r
###PCA
pred_collected_en_pca %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    719    297
##     offens  150    572
```

```r
pred_collected_rf_pca %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    763    164
##     offens  106    705
```

```r
pred_collected_knn_pca %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    867    595
##     offens    2    274
```

We can also look at the different measures of each of the model.

```r
pred_collected_en %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.792
##  2 kap                  binary         0.585
##  3 sens                 binary         0.804
##  4 spec                 binary         0.780
##  5 ppv                  binary         0.785
##  6 npv                  binary         0.800
```

```
##  7 mcc                  binary         0.585
##  8 j_index              binary         0.585
##  9 bal_accuracy         binary         0.792
## 10 detection_prevalence binary         0.512
## 11 precision            binary         0.785
## 12 recall               binary         0.804
## 13 f_meas               binary         0.795
```

```
pred_collected_rf %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.842
##  2 kap                  binary         0.684
##  3 sens                 binary         0.875
##  4 spec                 binary         0.809
##  5 ppv                  binary         0.821
##  6 npv                  binary         0.866
##  7 mcc                  binary         0.685
##  8 j_index              binary         0.684
##  9 bal_accuracy         binary         0.842
## 10 detection_prevalence binary         0.533
## 11 precision            binary         0.821
## 12 recall               binary         0.875
## 13 f_meas               binary         0.847
```

```
pred_collected_knn %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.651
##  2 kap                  binary         0.303
##  3 sens                 binary         0.992
##  4 spec                 binary         0.311
##  5 ppv                  binary         0.590
##  6 npv                  binary         0.975
##  7 mcc                  binary         0.413
##  8 j_index              binary         0.303
##  9 bal_accuracy         binary         0.651
## 10 detection_prevalence binary         0.841
## 11 precision            binary         0.590
## 12 recall               binary         0.992
## 13 f_meas               binary         0.740
```

```
##PCA

pred_collected_en_pca %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
```

```
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.743
##  2 kap                  binary         0.486
##  3 sens                 binary         0.827
##  4 spec                 binary         0.658
##  5 ppv                  binary         0.708
##  6 npv                  binary         0.792
##  7 mcc                  binary         0.493
##  8 j_index              binary         0.486
##  9 bal_accuracy         binary         0.743
## 10 detection_prevalence binary         0.585
## 11 precision            binary         0.708
## 12 recall               binary         0.827
## 13 f_meas               binary         0.763
```

```
pred_collected_rf_pca %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.845
##  2 kap                  binary         0.689
##  3 sens                 binary         0.878
##  4 spec                 binary         0.811
##  5 ppv                  binary         0.823
##  6 npv                  binary         0.869
##  7 mcc                  binary         0.691
##  8 j_index              binary         0.689
##  9 bal_accuracy         binary         0.845
## 10 detection_prevalence binary         0.533
## 11 precision            binary         0.823
## 12 recall               binary         0.878
## 13 f_meas               binary         0.850
```

```
pred_collected_knn_pca %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.657
##  2 kap                  binary         0.313
##  3 sens                 binary         0.998
##  4 spec                 binary         0.315
##  5 ppv                  binary         0.593
##  6 npv                  binary         0.993
##  7 mcc                  binary         0.428
##  8 j_index              binary         0.313
##  9 bal_accuracy         binary         0.657
## 10 detection_prevalence binary         0.841
## 11 precision            binary         0.593
## 12 recall               binary         0.998
## 13 f_meas               binary         0.744
```

We can see the best model is the Random forest model looking at the accuracy measure, where we get a measure of 0.8604924.

on test data

We now do the same predictions using the test data.

```
pred_collected_en_test <- tibble(
  truth = data_test_prep %>% pull(y),
  pred = fit_en %>% predict(new_data = data_test_prep) %>% pull(.pred_class),
  pred_prob = fit_en %>% predict(new_data = data_test_prep, type = "prob") %>% pull(.pred_offens),
  )

pred_collected_knn_test <- tibble(
  truth = data_test_prep %>% pull(y),
  pred = fit_knn %>% predict(new_data = data_test_prep) %>% pull(.pred_class),
  pred_prob = fit_knn %>% predict(new_data = data_test_prep, type = "prob") %>% pull(.pred_offens),
  )

pred_collected_rf_test <- tibble(
  truth = data_test_prep %>% pull(y),
  pred = fit_rf %>% predict(new_data = data_test_prep) %>% pull(.pred_class),
  pred_prob = fit_rf %>% predict(new_data = data_test_prep, type = "prob") %>% pull(.pred_offens),
  )


##PCA

pred_collected_en_test_pca <- tibble(
  truth = data_test_prep_pca %>% pull(y),
  pred = fit_en_pca %>% predict(new_data = data_test_prep_pca) %>% pull(.pred_class),
  pred_prob = fit_en_pca %>% predict(new_data = data_test_prep_pca, type = "prob") %>% pull(.pred_offens
  )

pred_collected_knn_test_pca <- tibble(
  truth = data_test_prep_pca %>% pull(y),
  pred = fit_knn_pca %>% predict(new_data = data_test_prep_pca) %>% pull(.pred_class),
  pred_prob = fit_knn_pca %>% predict(new_data = data_test_prep_pca, type = "prob") %>% pull(.pred_offe
  )

pred_collected_rf_test_pca <- tibble(
  truth = data_test_prep_pca %>% pull(y),
  pred = fit_rf_pca %>% predict(new_data = data_test_prep_pca) %>% pull(.pred_class),
  pred_prob = fit_rf_pca %>% predict(new_data = data_test_prep_pca, type = "prob") %>% pull(.pred_offens
  )
```

We again create confusion matrix for each model.

```
pred_collected_en_test %>% conf_mat(truth, pred)
```

```
##             Truth
## Prediction Hate offens
```

```
##     Hate    220    859
##     offens   45   2552
```

```
pred_collected_rf_test %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    216    783
##     offens   49   2628
```

```
pred_collected_knn_test %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    242   2666
##     offens   23    745
```

```
##PCA

pred_collected_en_test_pca %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    207   1061
##     offens   58   2350
```

```
pred_collected_knn_test_pca %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    229   1796
##     offens   36   1615
```

```
pred_collected_rf_test_pca %>% conf_mat(truth, pred)
```

```
##           Truth
## Prediction Hate offens
##     Hate    206    995
##     offens   59   2416
```

We can again look at the different measures of the models.

```
pred_collected_en_test %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.754
##  2 kap                  binary         0.239
```

```
##   3 sens                 binary         0.830
##   4 spec                 binary         0.748
##   5 ppv                  binary         0.204
##   6 npv                  binary         0.983
##   7 mcc                  binary         0.328
##   8 j_index              binary         0.578
##   9 bal_accuracy         binary         0.789
## 10 detection_prevalence binary         0.294
## 11 precision            binary         0.204
## 12 recall               binary         0.830
## 13 f_meas               binary         0.327
```

```
pred_collected_rf_test %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##   1 accuracy             binary         0.774
##   2 kap                  binary         0.257
##   3 sens                 binary         0.815
##   4 spec                 binary         0.770
##   5 ppv                  binary         0.216
##   6 npv                  binary         0.982
##   7 mcc                  binary         0.340
##   8 j_index              binary         0.586
##   9 bal_accuracy         binary         0.793
## 10 detection_prevalence binary         0.272
## 11 precision            binary         0.216
## 12 recall               binary         0.815
## 13 f_meas               binary         0.342
```

```
pred_collected_knn_test %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##   1 accuracy             binary        0.268
##   2 kap                  binary        0.0235
##   3 sens                 binary        0.913
##   4 spec                 binary        0.218
##   5 ppv                  binary        0.0832
##   6 npv                  binary        0.970
##   7 mcc                  binary        0.0837
##   8 j_index              binary        0.132
##   9 bal_accuracy         binary        0.566
## 10 detection_prevalence binary        0.791
## 11 precision            binary        0.0832
## 12 recall               binary        0.913
## 13 f_meas               binary        0.153
```

```
##PCA

pred_collected_en_test_pca %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.696
##  2 kap                  binary         0.171
##  3 sens                 binary         0.781
##  4 spec                 binary         0.689
##  5 ppv                  binary         0.163
##  6 npv                  binary         0.976
##  7 mcc                  binary         0.256
##  8 j_index              binary         0.470
##  9 bal_accuracy         binary         0.735
## 10 detection_prevalence binary         0.345
## 11 precision            binary         0.163
## 12 recall               binary         0.781
## 13 f_meas               binary         0.270
```

```
pred_collected_knn_test_pca %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary        0.502
##  2 kap                  binary        0.0831
##  3 sens                 binary        0.864
##  4 spec                 binary        0.473
##  5 ppv                  binary        0.113
##  6 npv                  binary        0.978
##  7 mcc                  binary        0.176
##  8 j_index              binary        0.338
##  9 bal_accuracy         binary        0.669
## 10 detection_prevalence binary        0.551
## 11 precision            binary        0.113
## 12 recall               binary        0.864
## 13 f_meas               binary        0.2
```

```
pred_collected_rf_test_pca %>% conf_mat(truth, pred) %>% summary()
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.713
##  2 kap                  binary         0.185
##  3 sens                 binary         0.777
##  4 spec                 binary         0.708
##  5 ppv                  binary         0.172
##  6 npv                  binary         0.976
##  7 mcc                  binary         0.268
##  8 j_index              binary         0.486
##  9 bal_accuracy         binary         0.743
## 10 detection_prevalence binary         0.327
## 11 precision            binary         0.172
## 12 recall               binary         0.777
## 13 f_meas               binary         0.281
```

On the testing data we see the best model is the elastic net model when looking at the accuracy measure..