Predicting Education Level From Writing

PSTAT 131

Julian Marks
Timothy Nguyen

December 21, 2017

1 Abstract

We set out to learn whether or not the words people use to describe themselves carried much information regarding their education level from a data mining standpoint. To this end, we have obtained a dataset containing anonymized user profile information from 59,946 San Francisco OKCupid users from June 2012. We employ the very popular text mining technique, Term Frequency - Inverse Document Frequency to transform users' answers to essay questions into data conducive to quantitative analysis. Then, we apply multiclass logistic regression and random forests classification techniques to predict their education levels, which were included in their profiles.

We found that the words with the most predictive power were those regarding education and those that implied lack of education, including a number of abbreviations and slang words. Through these variables we were able to construct a model that successfully predicted the class levels with 40-60% accuracy, which is 2-3 times better than the average guess, thus the attempts to create a useful predictor for education were successful.

- point about most predictive words - better or worse than guessing

2 Introduction

With the online populace rapidly growing in prominence and importance, electronic companies need innovations to keep their product on the foreground of expansion in order to keep up with the consequential effect of growing integration of technology and the internet in day to day life. OKCupid has not only created a product that brings dating to the electronic realm, a place where the younger generation find themselves more comfortable and accessible, but a place that stockpiles information on these people. This provides an avenue for data miners to effectively study the habits of many people using statistical methods. For example, the team behind OKCupid was able to find a

trend regarding the sort of messages and response a user receives based on their race; not to mention the intensive algorithmic construction that goes behind the matchmaking procedures for the primary site function. (Del Rey 2012)

The importance of data mining in a rapidly growing technocentric society is plainly visible. As machine learning algorithm development increases its potential, the ability for programmers to recognize and analyze attributes via programs will excel and prove vital when attempting to assess the behavior of others. This is especially true in the process of text mining where the most primary form of communication can be assessed, and with extensive training computers can be taught to recognize and realize as much of what is communicated through language as humans.

The data used is gathered entirely from OKCupid users in San Francisco, and it is based on personal response and evaluation. As such there could be bias when creating a profile, however this is still an effective data set; because of the nature of the text response questions and the desire to match two people for such intense cooperation, the text responses should act as an accurate avatar of that individual. Focus on users from San Francisco may not have as good predictive accuracy in other areas but within the bounds of San Francisco modelling and predicting this information will suffice. Using the data provided in the essay questions, we will address whether word choice is a significant factor in determining the level of education of any particular user. The words in all of the essays for each person were bagged, and their education level was separated into a number of bins due to the large variance in responses from the education variable.

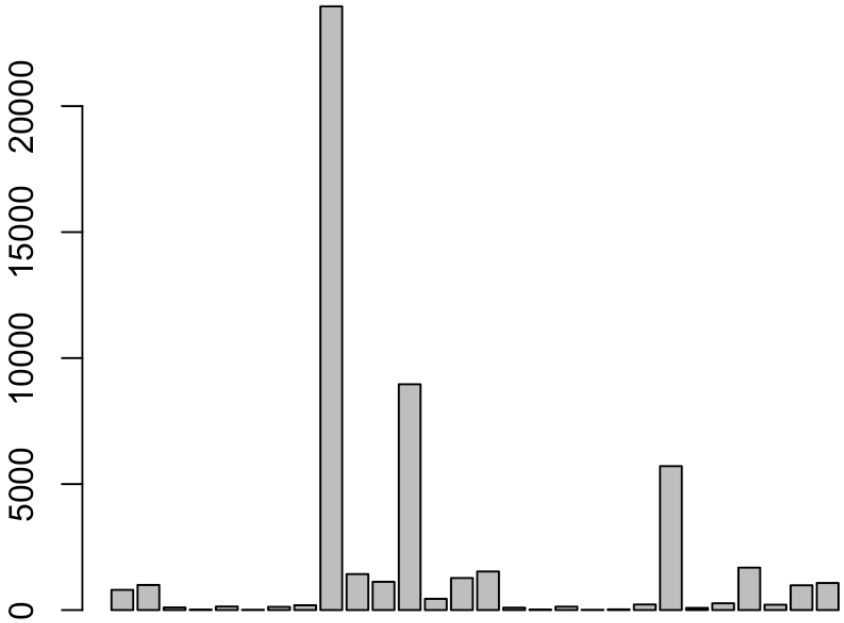
3 Data and Methods

Although our data ultimately boiled down to no more than two variables, text response and education level, both of them required substantial preprocessing. Our dependant variable originally contained too great a variety of categories for us to hope to successfully

predict. Untouched, the education level variable contained a total of 28 levels as shown here:

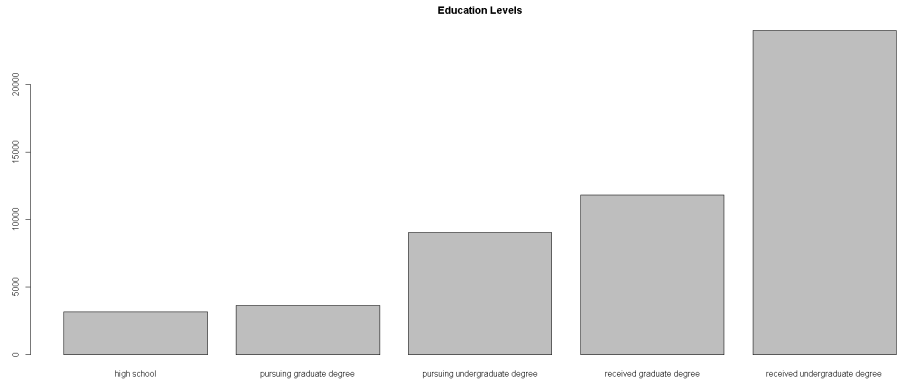
```
> unique(profiles$education)
 [1] "working on college/university"     "graduated from masters program"    "graduated from college/university"
 [4] "working on two-year college"       "graduated from high school"        "working on masters program"
 [7] "college/university"                "graduated from ph.d program"       "graduated from law school"
[10] "working on ph.d program"           "two-year college"                  "graduated from two-year college"
[13] "working on med school"             "dropped out of college/university" "graduated from med school"
[16] "dropped out of high school"        "working on high school"            "masters program"
[19] "dropped out of ph.d program"       "dropped out of two-year college"   "dropped out of med school"
[22] "high school"                       "working on law school"             "law school"
[25] "dropped out of masters program"    "ph.d program"                      "dropped out of law school"
[28] "med school"
```

We noticed that a number of the responses pertained to "space camp", but with no information on what this means there is no way of using it in our analysis. Hence we decided to remove all profiles with education level containing "space camp" specification Predicting among such a high number of categories is difficult enough, even with an even distribution. Here is a bar chart depicting the distribution among these education levels:

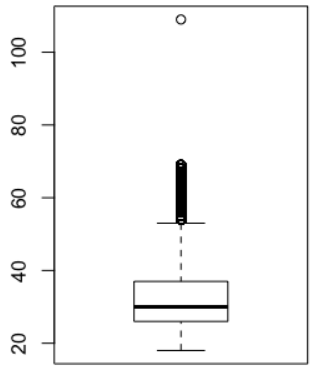


Without concerning ourselves with which education levels are the most prevalent, we can already see that the education levels are far from evenly distributed across users; thus, an already difficult problem has become more challenging. To address the problem of having too many categories to predict upon, we chose to bin the values such that they fell into one of five education levels: high school, pursuing undergraduate degree, received undergraduate degree, pursuing graduate degree, and received graduate degree. To ac-

complish this, we wrote a simple function that string-matched the original responses and mapped them to one of our designated categories. Here is the distribution of education levels after binning:



The image above depicts an uneven distribution which shows that a relatively high proportion of users have received degrees whether they be undergraduate degrees or graduate degrees. This makes sense considering the distribution of age among the users:



With a median age of 30 and mean age of 32, it is understandable that most of the users we are looking at have received their degrees and are no longer in school.

Just as we had to adjust our dependant variable, we also had to manipulate our predictive variable, the text. Earlier, we stated that our data boiled down to no more than two variables, but that is not completely true; our text data was originally split among

ten variables, one for each question. To simplify our text mining process, we decided to concatenate each user's responses to all of the questions to form a comprehensive body of text for each user. From there, we performed all of our text mining using the R package 'Quanteda'. Our first step was building a corpus from the set of texts we previously constructed by concatenating the responses. In the context of Quanteda, a corpus is defined as a collection of texts that includes document-level variables specific to each text, as well as meta-data for documents and for the collection as a whole.

Once we had our corpus, we aimed to quantify the text data by transforming it into a term frequency matrix. As the name implies, the matrix represents the text in the form of frequencies of each word in each document, effectively turning our text data into numbers more conducive to the machine learning techniques we are familiar with. We built the matrix using Quanteda's 'tfm()' function, which conveniently cleaned the text for us by removing non-text characters such as punctuation and numbers present in the responses. Additionally, we removed stop words, commonly used words that we expect to find in nearly all the responses therefore lacking discriminatory information, as they are defined by Quanteda. We also used word stemming, a technique that reduces words to their most basic form in hopes of aggregating derivationally related words with similar meanings. For example, the words "democratic" and "democratization" would be considered the same once stemmed. Most of the time, words that are related in this manner carry the same information in either form, so stemming our words and considering similar words as the same increases the significance of our predictors, the words. Then, we "trimmed" our matrix to keep only words that appeared in more than 1000 documents. This provided us with a matrix that contained less insignificant variables and was significantly less sparse, therefore easier to compute on. Next, we further refined our term frequency data by employing a technique called Term Frequency - Inverse Document Frequency (TF-IDF). The purpose of this technique is to give less weight to words which are more common

and therefore less discriminatory variables. To this end, the TF-IDF weight of a word is composed of two terms: (1) term frequency, which is computed as the number of times a word appears in a document, divided by the total number of words in that document, and (2) inverse document, computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears. These two components are multiplied together to produce the TF-IDF weight of a word. The last thing we did was center and scale the TF-IDF weights to help our model perform better.

Logistic Regression is a vital tool to use when looking for classified results. In this case there are a number of classes among which we are trying to predict, namely the education level of the users. We are looking to classify between 5 classes using our TF-IDF matrix as predictors, thus every unique word has some predictive power in discerning the education level of the person who wrote it. This model analyzes their word choice to create a linear classifier that returns probabilities that a user is each of the response classes. Then the best prediction for that model is the class that returns the highest predicted probability. Because our data is sparse, we elected to use the package LibLineaR, a package for linear classification conducive to sparse data. This package provides a function that performs logistic regression, and we have chosen the L2 Loss Function to regulate our model. The L2 refers to using Least Squares Error to estimate variance in the model, opposed to estimation using Least Absolute Deviation (L1) which is more computationally intensive on sparse data. We used a for loop to loop through four cost values: 100, 1, 0.01, and 0.001. In each iteration of the loop, we used 3-fold cross validation to check the accuracy achieved by each cost value. Ultimately, the best cost value was 0.001, and that is the value we used in training our final logistic regression model.

Random Forests were also implemented on this data set. This model is good for this sort of data due to the large number of observations and predictions. Random Forests

create a large number of trees by selecting a subset of predictors at each node, which allows a great amount of the predictors to be utilized among all of the trees. New observations are sent through all of the trees and By using a random forest model strong predictors can be filtered out, and with the predictors as the TF-IDF of each word the strongest predicting words can be identified in addition to creating a number of models which can check and balance each other, even when some trees miss the important variables. It also gives all of the predictors a chance to show their significance; by being thorough and checking the many predictors that are available, not only the strongest predictors will be included but all words that carry some amount of weight.

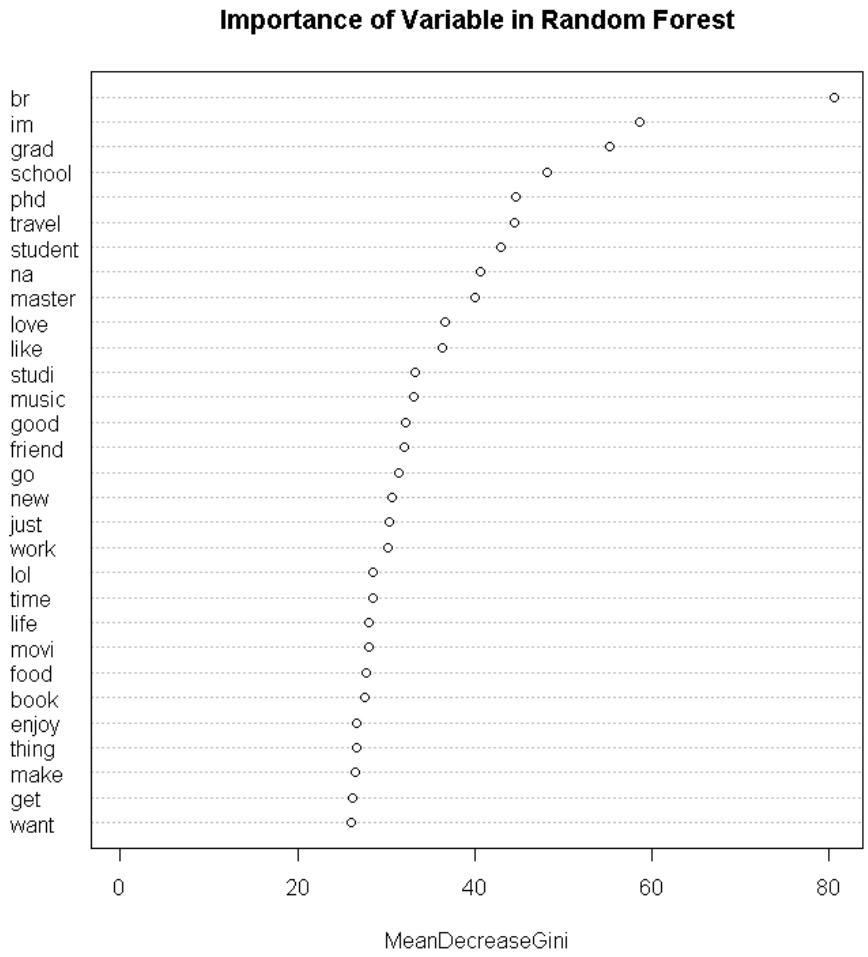
4 Results

The output from the random forest shows an average accuracy of about 42%, which is twice as good as random guess when predicting for 5 classifications. This being the average, the random forest actually predicts some classes with higher accuracy than others, reaching as low as 25% for those pursuing an undergraduate degree to 62% for those classified as high school. The accuracy may not be high, but this predictor is twice as good at classification of education as an average guesser, which in the case of predicting between 5 categories is expected to achieve .

```
> gen_accuracy(cfm_rf)
Accuracy for high school: 0.6253776
Accuracy for pursuing undergraduate degree: 0.2625608
Accuracy for received undergraduate degree: 0.281399
Accuracy for pursuing graduate degree: 0.5222734
Accuracy for received graduate degree: 0.4305772
Average accuracy:  0.4244376
```

```
> print(cfm_rf)
                              Test
True                           high school pursuing graduate degree pursuing undergraduate degree received graduate degree received undergraduate degree
  high school                          414                       80                           211                       76                           114
  pursuing graduate degree              31                      340                            83                      124                            96
  pursuing undergraduate degree        114                       52                           162                       39                            68
  received graduate degree              44                      102                            62                      276                           174
  received undergraduate degree         59                       77                            99                      126                           177
```

The importance of variables show that significant predictors have to do with school or education, and include shorthand typing abbreviations. This means that the most drastic variables from both ends, both the most and least intellectual word, have the highest predictive significance, and this is reflected in the accuracy rates



Our best logistic regression model performed no better than our random forest model, achieving an average accuracy of 42%:

```
Accuracy for high school: 0.6022013
Accuracy for pursuing undergraduate degree: 0.3015625
Accuracy for received undergraduate degree: 0.3045526
Accuracy for pursuing graduate degree: 0.4881517
Accuracy for received graduate degree: 0.4357798
Average accuracy:  0.4264496
```

	high school	pursuing graduate degree	pursuing undergraduate degree	received graduate degree	received undergraduate degree
high school	383	60	227	60	86
pursuing graduate degree	40	309	86	129	82
pursuing undergraduate degree	125	72	193	58	91
received graduate degree	34	113	52	285	184
received undergraduate degree	54	79	82	122	194

However, the logistic regression model has the added benefit of increased interpretability. Here, we show the top 20 most predictive words for each predicted category:

```
Top features for 'high school':
 [1] "im"         "transfer"   "gamer"      "pool"       "know"       "metal"      "shi"        "high"       "anarchi"    "friendship"
[11] "go"         "money"      "video"      "lol"        "music"      "judgment"   "fish"       "misfit"     "that"       "weed"
Top features for 'pursuing undergraduate degree':
 [1] "major"       "studi"      "sfsu"       "degre"      "internship" "academi"    "semest"     "integr"     "br"         "bachelor"
[11] "photographi" "homework"   "across"     "firefli"    "awkward"    "bacon"      "student"    "quick"      "creepi"     "uc"
Top features for 'received undergraduate degree':
 [1] "graduat"   "weekend"   "br"        "compani"   "barbara"   "enjoy"     "fidel"     "organ"     "lunch"     "citi"      "australia"
[12] "whiskey"   "golf"      "ramen"     "dragon"    "project"   "hbo"       "cohen"     "junk"      "boot"
Top features for 'pursuing graduate degree':
 [1] "grad"      "phd"       "master"    "ph.d"      "mba"       "law"       "student"   "school"    "studi"     "research"  "graduat"
[12] "scienc"    "berkeley"  "undergrad" "program"   "pursu"     "clinic"    "stanford"  "medicin"   "doctor"
Top features for 'received graduate degree':
 [1] "lawyer"     "scientist"  "dinner"     "us"         "passport"   "yoga"       "popular"    "check"      "t-shirt"    "grate"
[11] "job"        "startup"    "practic"    "oyster"     "indian"     "travel"     "profession" "ski"        "long"       "includ"
```

Here is where we found great interest; while the model did not predict with superb accuracy, it did extract some meaningful information from the text. We can see that users who are currently students (i.e., pursuing undergraduate/graduate school) are more inclined to use words pertaining to school, likely because it is what most of their life revolves around. Also interesting is that users in the "received graduate degree" category are inclined to use words such as "lawyer", "scientist", "job", "startup", and "profession".

5 Conclusion

Both of our predictive models were able to come up with correct classification rates of approximately 42%. In this sense, they double the accuracy of random guesses, so we are satisfied with our results. Some difficulty is the incredible overlap that exists within language, especially English. The most common words used to communicate is known by all who know the basics of the language, and often grammar is a strong indication of knowledge of a language. In the future, a large-scale project would include the analysis of features involving syntax, word length, and other grammatical details, though this would add an immense layer of difficulty in the programming aspect. Unfortunately this could be extremely crucial to predicting levels of education.

Though both the logistic regression and random forest models had about a 42% average accuracy rate, the random forest was able to predict some of the classes better than the logistic regression model. In particular, the random forest had higher prediction rates for the High School and Pursuing Grad Degree bins; this could be reflective of its ability to distinguish the important words, which include both extremely basic and extremely intellectual words that would be used by people of those education levels respectively.

Finally, based on the top features we see from the logistic regression model, we would be interested in reframing this project as a binary classification problem in which we attempt to predict if a user is a student or not. We would expect that we can do so to a high degree of success.

6 References

2017. "Working with Text Data", *Scikit-learn.*

URL: http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

Abramovich, Giselle. 2012. "How OKCupid Built a Data-First Brand", *DigiDay.*

URL: https://digiday.com/marketing/how-okcupid-built-its-brand-on-data/

"Getting Started with quanteda", *CRAN.*

URL: https://cran.r-project.org/web/packages/quanteda/vignettes/quickstart.html

Helleputte, Thibault and Gramme, Pierre. 2013. "Linear Predictive Models Based on the 'LIBLINEAR' C/C++ Library", *CRAN*

URL: https://cran.r-project.org/web/packages/LiblineaR/LiblineaR.pdf

7 Appendix

```
# GOAL: CAN WE PREDICT EDUCATION LEVEL BASED ON WRITING STYLE?
# DATA: https://github.com/rudeboybert/JSE_OkCupid
# methods used: bag of words analysis


# Libraries ----
library(tidyverse)
library(quanteda)
library(readtext)
library(stringr)
library(topicmodels)
library(tidytext)
library(caret)
library(LiblineaR)
library(randomForest)

# Constants ----
set.seed(1)

DATA_DIR <- '/Users/timmy/Documents/school/year-3/pstat131/
    okcupid'

# EDUCATION LEVEL BINS
# rules:
```

```
#   (1) if it doesnt say "graduated" in it, we assumed its in
    progress
#   (2) we considered "dropped out" to count as "pursuing" bc
    the person did some of the work
#   (3) we lumped all of the "two-year college" education levels
     in with highschool
#       except for "graduated from two-year college" which goes
    in "pursuing UG"
HS_BIN <- c("graduated from high school", "high school", "
    working on two-year college",
            "dropped out of high school", "working on high
                school", "two-year college",
            "dropped out of two-year college")
P_UG_BIN <- c("working on college/university", "college/
    university",
              "dropped out of college/university", "graduated
                  from two-year college")
R_UG_BIN <- c("graduated from college/university")
P_G_BIN <- c("working on masters program", "working on ph.d
    program", "working on med school",
             "masters program", "dropped out of med school", "
                 dropped out of ph.d program",
             "working on law school", "law school", "dropped out
                 of masters program",
             "ph.d program", "dropped out of law school", "med
                 school")
```

```
R_G_BIN <- c("graduated from masters program", "graduated from
    ph.d program",
                "graduated from law school", "graduated from med
                    school")
SC_BIN <- c("space camp", "dropped out of space camp", "working
    on space camp",
                "graduated from space camp")
BINS <- c("high school", "pursuing undergraduate degree", "
    received undergraduate degree",
              "pursuing graduate degree", "received graduate degree"
                  )

# Functions ----

# function for mapping education levels into one of the
    following bins:
#   [highschool, pursuing undergraduate degree, received
    undergraduate degree,
#   pursuing graduate degree, received graduate degree, other]
edu_bin <- function(s) {
  if (s %in% HS_BIN) {
    "high school"
  } else if (s %in% P_UG_BIN) {
    "pursuing undergraduate degree"
  } else if (s %in% R_UG_BIN) {
    "received undergraduate degree"
```

```
  } else if (s %in% P_G_BIN) {
    "pursuing graduate degree"
  } else if (s %in% R_G_BIN) {
    "received graduate degree"
  }
}

accuracy <- function(cfm, bin) {
  cfm[bin, bin]/sum(cfm[,bin])
}

gen_accuracy <- function(cfm) {
  all_acc <- c()
  for (bin in BINS) {
    acc <- accuracy(cfm, bin)
    all_acc <- c(all_acc, acc)
    cat("Accuracy for ", bin, ": ", acc, "\n", sep="")
  }
  cat("Average accuracy: ", mean(all_acc))
}

top_features <- function(model, edu, n = 20) {
  names(sort(model$W[edu,], T)[1:n])
}

gen_top_features <- function(model, n = 20) {
```

```
  for (bin in BINS) {
    cat("Top features for '", bin, "':\n", sep="")
    print(top_features(model, bin, n))
  }
}

# Main ----
# Paths
profiles_path <- file.path(DATA_DIR, 'profiles.csv')

# Loading data
original_profiles <- read_csv(profiles_path)
profiles <- original_profiles

### EDUCATION LEVEL BINNING
# there are 32 levels of education right now, we want fewer
  levels, so we'll bin them
unique(profiles$education)

# there are 6628 rows with missing values for education
sum(is.na(profiles$education))
# there are 1683 rows with education level involving "space camp
  ", which we doesn't tell us much
sum(profiles$education %in% SC_BIN)
# we remove these rows bc it's only ~15% of our data, still
  leaving us with 52k rows
```

```
profiles <- profiles %>% filter(!education %in% c(NA, SC_BIN))

# now we use previously defined function to put education level
    into bins
education <- sapply(profiles$education, edu_bin)
# frequency counts for education level
table(education)
# barplot of education level freqs
barplot(table(education))
# notice: lots of ppl with degrees received...
# checking age, makes sense considering median age 30 and mean
    age 32
summary(profiles$age)


### FORMATTING TEXT
# concatenating answers from all questions to be one body of
    text
text <- paste(profiles$essay0, profiles$essay1, profiles$essay2,
              profiles$essay3, profiles$essay4, profiles$essay5,
              profiles$essay6, profiles$essay7, profiles$essay8,
              profiles$essay9, sep = " ")

# constructing new data frame with a column denoting sex of
    profile and
#   another containing their comprehensive writing
```

```
df <- data.frame(text, stringsAsFactors = F)
df$doc_id <- seq(dim(df)[1])

# building a corpus
my_corpus <- corpus(df)


### MODELING
# creating document frquency matrix
my_dfm <- dfm(my_corpus, remove = stopwords("english"), stem =
    TRUE,
                  remove_punct = TRUE, remove_numbers = TRUE)
# removing features that occur in fewer than 2 documents
(my_dfm <- dfm_trim(my_dfm, min_docfreq = 1000, verbose=TRUE))

# using text frequency - inverse document frequency weighting
my_tfidf <- tfidf(my_dfm, normalize = TRUE) %>%
   as.matrix()

# sampling (creating vector of indices)
train_ind <- sample(seq_len(nrow(df)), size = nrow(df) * .8)

# using indices to pslit data into train and test
tr_X <- my_tfidf[train_ind,]
tr_Y <- education[train_ind]
te_X <- my_tfidf[-train_ind,]
```

```
te_Y <- education[-train_ind]
dim(tr_X)
dim(te_X)

# training first model
tr_X_s <- scale(tr_X, center=TRUE, scale = TRUE)

start.time <- Sys.time()
m <- LiblineaR(data = tr_X_s, target = tr_Y, type = 7, bias = 1,
    verbose = FALSE)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken

te_X_s <- scale(te_X,attr(tr_X_s,"scaled:center"),attr(tr_X_s,"
   scaled:scale"))
p <- predict(m, te_X_s, decisionValues = TRUE)

cfm_init_logreg <- table(p$predictions,te_Y)
print(cfm_init_logreg)
gen_accuracy(cfm_init_logreg)

# poor results probably due to imblanced classes
# imbalanced classes:
# https://machinelearningmastery.com/tactics-to-combat-
   imbalanced-classes-in-your-machine-learning-dataset/
```

```
# DOWNSAMPLING
df_ds <- downSample(df, as.factor(education), yname ='education'
    )
education_ds <- df_ds$education
df_ds <- df_ds %>% select(-education)

my_corpus_ds <- corpus(df_ds)

# making document frequency matrix from downsampled data
my_dfm_ds <- dfm(my_corpus_ds, remove = stopwords("english"),
    stem = TRUE,
                     remove_punct = TRUE, remove_numbers = TRUE)
# removing features that occur in fewer than 100 documents
(my_dfm_ds <- dfm_trim(my_dfm_ds, min_docfreq = 100, verbose=
    TRUE))

# using text frequency - inverse document frequency weighting on
     downsampled dfm
my_tfidf_ds <- tfidf(my_dfm_ds, normalize = TRUE) %>%
   as.matrix()

# splitting downsampled data
train_ind <- sample(seq_len(nrow(df_ds)), size = nrow(df_ds) *
    .8)

tr_X_ds <- my_tfidf_ds[train_ind,]
```

```
tr_Y_ds <- education_ds[train_ind]
te_X_ds <- my_tfidf_ds[-train_ind,]
te_Y_ds <- education_ds[-train_ind]
dim(tr_X_ds)
dim(te_X_ds)

# scaling downsampled data
tr_X_ds_s <- scale(tr_X_ds, center = TRUE, scale = TRUE)
te_X_ds_s <- scale(te_X_ds, center = TRUE, scale = TRUE)

# training RF
start.time <- Sys.time()
rf1 <- randomForest(tr_X_ds_s, y = tr_Y_ds, ntree = 100)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken

p_rf_ds <- predict(rf1, te_X_ds_s)
# confusion matrix
cfm_rf <- table(p_rf_ds,te_Y_ds)
print(cfm_rf)
# outputting accuracy
gen_accuracy(cfm_rf)

# trying multiple cost values
costs <- c(100, 1, 0.01, 0.001)
```

```
best_cost <- NA
best_acc <- 0
for(cost in costs) {
  start.time <- Sys.time()
  acc <- LiblineaR(data=tr_X_ds_s,target=tr_Y_ds,type=7,cost=
      cost,bias=1,cross=3,verbose=FALSE)
  end.time <- Sys.time()
  time.taken <- end.time - start.time
  cat("time taken:", time.taken, sep = " ")
  cat("Results for C=", cost, " : ", acc, "accuracy.\n", sep="")
  if(acc>best_acc){
    best_cost=cost
    best_acc=acc
  }
}

cat("Best cost is:",best_cost,"\n")
cat("Best accuracy is:",best_acc,"\n")

# Re-train best model with best cost value.
best_lr_model <- LiblineaR(data=tr_X_ds_s,target=tr_Y_ds,type=7,
    cost=best_cost,bias=1,verbose=FALSE)
try_lr_model <- LiblineaR(data=tr_X_ds_s,target=tr_Y_ds,type=7,
    cost=0.001,bias=1,verbose=FALSE)
# Scale the test data
te_X_ds_s <- scale(te_X_ds,attr(tr_X_ds_s,"scaled:center"),attr(
```

```
    tr_X_ds_s,"scaled:scale"))
# Make prediction

p_lr_ds=predict(best_lr_model,te_X_ds_s,proba=TRUE,
    decisionValues=TRUE)
cfm_lr <- table(p_lr_ds$predictions,te_Y_ds)
print(cfm_lr)
# outputting accuracy
gen_accuracy(cfm_lr)
gen_top_features(best_lr_model)

p_lr_ds2=predict(try_lr_model,te_X_ds_s,proba=TRUE,
    decisionValues=TRUE)
cfm_lr2 <- table(p_lr_ds2$predictions,te_Y_ds)
print(cfm_lr2)
# outputting accuracy
gen_accuracy(cfm_lr2)
gen_top_features(try_lr_model)
```