DSP_word2vec_ds1

Melanie Weissenboeck

2022-11-28

Laden von Bibliotheken und Daten

```r
library("xlsx")
library(word2vec)
library(udpipe)
library(SnowballC)
library(ggplot2)
library(tm)
library(tidytext)
library(tidyr)
library(mlbench)
library(e1071)
library(caret)
library(class)
library(readr)
library(gmodels)
library(neuralnet)
```

Vorverarbeiten der Texte

Bereinigen der Texte

```r
# Funktion fuer Text Bereinigung
ds1$ANF_BESCHREIBUNG <- txt_clean_word2vec(ds1$ANF_BESCHREIBUNG,
                                           ascii = FALSE, alpha = TRUE,
                                           tolower = TRUE, trim = TRUE)
```

Erstellen einer Worteinbettung

```r
# Modell trainieren fuer Einbettung
model_ds1 <- word2vec(ds1$ANF_BESCHREIBUNG, dim = 10, iter = 15)
embedding_ds1 <- as.matrix(model_ds1)
```

```r
# Dimension der Einbettung
dim(embedding_ds1)
```

```
## [1] 632  10
```

Generieren von numerischen Prädiktoren

```
# aufteilen der Texte in einzelne Token
ds1$token <- tokenizers::tokenize_words(ds1$ANF_BESCHREIBUNG)
```

```
# Vektor der Laenge 10 fuer jedes Dokument
features <- matrix(nrow = 0, ncol = 10)
for (i in (1:length(ds1$ANF_BESCHREIBUNG))){
  vec_doc1 <- doc2vec(model_ds1, ds1$token[1][[1]][i], split = " ")
  features <- rbind(features, vec_doc1)
}
```

Zusammenführen mit anderen Prädiktoren

```
features <- as.data.frame(features)
ds1_all <- cbind(ds1, features)
```

```
ds1_all <- as.data.frame(ds1_all)
```

```
df <- ds1_all[ , c(4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16, 17, 18)]
df[is.na(df)] <- 0
head(df)
```

```
##   ANF_RISIKO TF_ABDECKUNG AKT_RES_STATUS AKT_RES_RELEASE          V1
## 1     mittel         16.6             OK             21x  1.93171088
## 2     mittel         16.7             OK             21x  0.28772104
## 3     gering         50.0             OK             21x  0.04097215
## 4     gering        100.0             OK             21x -0.64427375
## 5     gering        100.0             OK           22.10  1.89069740
## 6       hoch         20.0             OK             21x  0.04097215
##            V2        V3        V4         V6         V7         V8         V9
## 1 -0.54896788 0.3526800 0.9523733 -0.8342013 -0.3484041 -1.2318031  0.8534454
## 2 -0.09150264 1.3737816 0.8976325 -1.0328456 -0.7035472 -0.5623689  2.0201454
## 3 -1.08037693 1.3427004 1.3634826 -0.2684955 -0.5630286 -0.8838432  0.7237027
## 4  0.89061319 1.3380111 0.1161471  0.7969690  0.1329895 -0.2306758  2.0224008
## 5 -0.52412681 0.6002749 1.5281878 -0.2464703 -0.2363677 -0.2449340 -0.6110806
## 6 -1.08037693 1.3427004 1.3634826 -0.2684955 -0.5630286 -0.8838432  0.7237027
##         V10
## 1 0.2247278
## 2 0.9437121
## 3 0.9114979
## 4 1.3846900
## 5 0.4560113
## 6 0.9114979
```

Normalisieren numerischer Spalten

```
set.seed(1234)

# definiere normalisierungsfunktion
min_max_norm <- function(x) {
(x - min(x)) / (max(x) - min(x))
}
# alle spalten normalisieren
```

```r
df[, 5:13] <- as.data.frame(lapply(df[, 5:13], min_max_norm))
df[2] <- as.data.frame(lapply(df[2], min_max_norm))

df$ANF_RISIKO <- as.factor(df$ANF_RISIKO)
df$AKT_RES_STATUS <- as.factor(df$AKT_RES_STATUS)
df$AKT_RES_RELEASE <- as.factor(df$AKT_RES_RELEASE)
summary(df)
```

```
##   ANF_RISIKO   TF_ABDECKUNG    AKT_RES_STATUS AKT_RES_RELEASE       V1
##  gering:158   Min.   :0.0000   FAILED: 12     21x     : 30    Min.   :0.0000
##  hoch  : 84   1st Qu.:0.5000   OK    :359     22.10   :132    1st Qu.:0.3228
##  mittel:136   Median :1.0000   OPEN  :  7     22.20   :129    Median :0.3228
##               Mean   :0.8056                  22.30   :  3    Mean   :0.3285
##               3rd Qu.:1.0000                  OLDERT21: 84    3rd Qu.:0.3228
##               Max.   :1.0000                                  Max.   :1.0000
##        V2               V3               V4               V6
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.5136   1st Qu.:0.2073   1st Qu.:0.3759   1st Qu.:0.3491
##  Median :0.5136   Median :0.2073   Median :0.3759   Median :0.3491
##  Mean   :0.5184   Mean   :0.2430   Mean   :0.3819   Mean   :0.3642
##  3rd Qu.:0.5136   3rd Qu.:0.2073   3rd Qu.:0.3759   3rd Qu.:0.3491
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##        V7               V8               V9              V10
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.9255   1st Qu.:0.7705   1st Qu.:0.3526   1st Qu.:0.2675
##  Median :0.9255   Median :0.7705   Median :0.3526   Median :0.2675
##  Mean   :0.9009   Mean   :0.7602   Mean   :0.3766   Mean   :0.2813
##  3rd Qu.:0.9255   3rd Qu.:0.7705   3rd Qu.:0.3526   3rd Qu.:0.2675
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Klassifikation

Erstellen von Train- / Test-Split

```r
# partition erstellen
part <- createDataPartition(df$ANF_RISIKO, times = 1, p = 0.80)
# extract training set
X_train <- df[part$Resample1, ]
# extract testing set
X_test <- df[-part$Resample1, ]
# extract target
y_train <- df[part$Resample1, 1]
y_test <- df[-part$Resample1, 1]
```

NaiveBayes Klassifikation

```r
model_nb = naiveBayes(ANF_RISIKO ~ ., data = X_train)
```

```r
pred_nb <- predict(model_nb, X_test)
mat.nb <- confusionMatrix(pred_nb, X_test$ANF_RISIKO, mode = "prec_recall")
mat.nb
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction gering hoch mittel
##     gering     26    6     16
##     hoch        2    6      6
##     mittel      3    4      5
##
## Overall Statistics
##
##                Accuracy : 0.5
##                  95% CI : (0.3814, 0.6186)
##     No Information Rate : 0.4189
##     P-Value [Acc > NIR] : 0.09807
##
##                   Kappa : 0.2041
##
##  Mcnemar's Test P-Value : 0.01023
##
## Statistics by Class:
##
##                      Class: gering Class: hoch Class: mittel
## Precision                   0.5417     0.42857       0.41667
## Recall                      0.8387     0.37500       0.18519
## F1                          0.6582     0.40000       0.25641
## Prevalence                  0.4189     0.21622       0.36486
## Detection Rate              0.3514     0.08108       0.06757
## Detection Prevalence        0.6486     0.18919       0.16216
## Balanced Accuracy           0.6635     0.61853       0.51812
```

KNN Klassifikation

```
model_knn <- train(ANF_RISIKO ~ ., data = X_train, "knn",
trControl = trainControl(method = "cv", number = 5))
```

```
pred_knn <- predict(model_knn, X_test, type = "raw")
mat.knn <- confusionMatrix(pred_knn, X_test$ANF_RISIKO, mode = "prec_recall")
mat.knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction gering hoch mittel
##     gering     27    7     16
##     hoch        0    6      1
##     mittel      4    3     10
##
## Overall Statistics
##
##                Accuracy : 0.5811
##                  95% CI : (0.4606, 0.6949)
##     No Information Rate : 0.4189
##     P-Value [Acc > NIR] : 0.003584
##
##                   Kappa : 0.3162
```

```
##
##  Mcnemar's Test P-Value : 0.001653
##
## Statistics by Class:
##
##                      Class: gering Class: hoch Class: mittel
## Precision                   0.5400     0.85714        0.5882
## Recall                      0.8710     0.37500        0.3704
## F1                          0.6667     0.52174        0.4545
## Prevalence                  0.4189     0.21622        0.3649
## Detection Rate              0.3649     0.08108        0.1351
## Detection Prevalence        0.6757     0.09459        0.2297
## Balanced Accuracy           0.6680     0.67888        0.6107
```