DSP_word2vec_ds3

Melanie Weissenboeck

2022-11-28

Laden von Bibliotheken und Daten

```
library("xlsx")
library(word2vec)
library(udpipe)
library(SnowballC)
library(ggplot2)
library(tm)
library(wordcloud)
library(tidytext)
library(tidyr)
library(mlbench)
library(e1071)
library(caret)
library(class)
```

Vorverarbeiten der Texte

Bereinigen der Texte

```
# Funktion fuer Text Bereinigung
ds3$ANF_BESCHREIBUNG <- txt_clean_word2vec(ds3$ANF_BESCHREIBUNG,
                                           ascii = FALSE, alpha = TRUE,
                                           tolower = TRUE, trim = TRUE)
```

Erstellen einer Worteinbettung

```
# Modell trainieren fuer Einbettung
model_ds3 <- word2vec(ds3$ANF_BESCHREIBUNG, dim = 10, iter = 15)
embedding_ds3 <- as.matrix(model_ds3)
```

```
# Dimension der Einbettung
dim(embedding_ds3)
```

```
## [1] 1391   10
```

Generieren von numerischen Prädiktoren

```
# # aufteilen der Texte in einzelne Token
ds3$token <- tokenizers::tokenize_words(ds3$ANF_BESCHREIBUNG)
```

```
# Vektor der Laenge 10 fuer jedes Dokument
features3 <- matrix(nrow = 0, ncol = 10)
for (i in (1:length(ds3$ANF_BESCHREIBUNG))){
  vec_doc1 <- doc2vec(model_ds3, ds3$token[1][[1]][i], split = " ")
  features3 <- rbind(features3, vec_doc1)
}
```

Zusammenführen mit anderen Prädiktoren

```
features3 <- as.data.frame(features3)
ds3_all <- cbind(ds3, features3)
ds3_all <- as.data.frame(ds3_all)
```

```
df <- ds3_all[ , c(6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 20)]
df[is.na(df)] <- 0
head(df)
```

```
##   ANF_RISIKO TF_ABDECKUNG AKT_RES_STATUS AKT_RES_RELEASE          V1
## 1     mittel        50.00             OK           22.30  1.81680287
## 2       hoch        50.00             OK           22.10  1.58138892
## 3       hoch        11.11             OK           22.10  1.51066036
## 4       hoch        50.00             OK           22.10  1.47084883
## 5       hoch        33.33             OK           22.10 -0.03306465
## 6     mittel        25.00             OK           22.10  0.67022652
##             V2         V3         V4         V6          V7         V8
## 1 -0.625880467 -0.8299286  1.2958461 -1.2824979  0.53619717  0.9516912
## 2 -0.748516666 -0.7544880  1.9151982 -0.7844844  0.06968573  1.1566638
## 3  0.036573360  0.3932653  1.9043160 -1.0189953 -0.51803669 -1.1585179
## 4  1.683262502 -0.3067315 -0.7762614  0.2065767 -0.06719837 -1.1157499
## 5  0.622790335  0.7612282  0.6083712 -1.7842711 -0.96411103  0.3981877
## 6  0.008048816  0.1555377  1.1388404 -0.9047653 -0.97472293  0.8141601
##            V9         V10
## 1 -0.35647663 -0.53895776
## 2 -0.82639836  0.07462885
## 3 -0.04000562  1.08260089
## 4 -1.06768932 -0.30208120
## 5  0.62528849 -1.48334753
## 6 -0.84430814 -2.18131948
```

Normalisieren numerischer Spalten

```
set.seed(1234)

# definiere normalisierungsfunktion
min_max_norm <- function(x) {
(x - min(x)) / (max(x) - min(x))
}
# alle spalten normalisieren
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:241   Min.   :0.0000   FAILED:  48    21x     :  48   Min.   :0.0000
##  hoch  :540   1st Qu.:0.0667   OK    :1395    22.10   :1081   1st Qu.:0.3810
##  mittel:665   Median :0.1429   OPEN  :   3    22.20   :  12   Median :0.3810
##               Mean   :0.2320                   22.30   : 302   Mean   :0.3883
##               3rd Qu.:0.3333                   OLDERT21:   3   3rd Qu.:0.3810
##               Max.   :1.0000                                   Max.   :1.0000
##        V2               V3               V4               V6
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.5220   1st Qu.:0.4979   1st Qu.:0.3925   1st Qu.:0.6427
##  Median :0.5220   Median :0.4979   Median :0.3925   Median :0.6427
##  Mean   :0.5258   Mean   :0.4996   Mean   :0.3994   Mean   :0.6357
##  3rd Qu.:0.5220   3rd Qu.:0.4979   3rd Qu.:0.3925   3rd Qu.:0.6427
##  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.:0.5967   1st Qu.:0.4424   1st Qu.:0.5135   1st Qu.:0.5266
##  Median :0.5967   Median :0.4424   Median :0.5135   Median :0.5266
##  Mean   :0.5964   Mean   :0.4429   Mean   :0.5134   Mean   :0.5263
##  3rd Qu.:0.5967   3rd Qu.:0.4424   3rd Qu.:0.5135   3rd Qu.:0.5266
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Klassifikation

Erstellen von Train- / Test-Split

```
# 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

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

```
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      10    3      3
##     hoch        36  102    117
```

```
##     mittel      2    3     13
##
## Overall Statistics
##
##                Accuracy : 0.4325
##                  95% CI : (0.3746, 0.4918)
##     No Information Rate : 0.4602
##     P-Value [Acc > NIR] : 0.8421
##
##                   Kappa : 0.1027
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: gering Class: hoch Class: mittel
## Precision                  0.62500      0.4000       0.72222
## Recall                     0.20833      0.9444       0.09774
## F1                         0.31250      0.5620       0.17219
## Prevalence                 0.16609      0.3737       0.46021
## Detection Rate             0.03460      0.3529       0.04498
## Detection Prevalence       0.05536      0.8824       0.06228
## Balanced Accuracy          0.59172      0.5496       0.53285
```

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     21   11      9
##     hoch       10   54      8
##     mittel     17   43    116
##
## Overall Statistics
##
##                Accuracy : 0.6609
##                  95% CI : (0.6032, 0.7153)
##     No Information Rate : 0.4602
##     P-Value [Acc > NIR] : 5.052e-12
##
##                   Kappa : 0.4377
##
##  Mcnemar's Test P-Value : 7.391e-06
##
## Statistics by Class:
##
```

```
##                      Class: gering Class: hoch Class: mittel
## Precision                  0.51220      0.7500        0.6591
## Recall                     0.43750      0.5000        0.8722
## F1                         0.47191      0.6000        0.7508
## Prevalence                 0.16609      0.3737        0.4602
## Detection Rate             0.07266      0.1869        0.4014
## Detection Prevalence       0.14187      0.2491        0.6090
## Balanced Accuracy          0.67726      0.7003        0.7438
```