DSP_tfidf_ds2

Melanie Weissenboeck

2022-10-16

Laden von Bibliotheken und Daten

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

Vorverarbeiten der Texte

Erstellen eines Korpus

Am Beginn der Vorverarbeitung wird zum `ds2` ein Wortkorpus erstellt. Dazu wird das `tm`-Package verwendet. Als Parameter wird angegeben, dass es sich um deutsche Texte handelt.

```
# CREATING CORPUS
# Corpus, VCorpus or SimpleCorpus -> SimpleCorpus
corp_ds2 = SimpleCorpus(VectorSource(ds2$ANF_BESCHREIBUNG), control = list(language = "de"))
corp_ds2
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 3121
```

Entfernen von störenden Zeichen

Es werden alle numerischen Zeichen aus den Texten entfernt. Zur Veranschaulichung wird ein Beispiel vor und nach dem Verarbeitungsschritt ausgegeben.

```
# REMOVING NUMBERS
corp_ds2 = tm_map(corp_ds2, removeNumbers)
```

Nach den numerischen Zeichen werden auch die Satzzeichen entfernt.

```
# REMOVE PUNCTUATION
corp_ds2 = tm_map(corp_ds2, removePunctuation)
```

Leerzeichen und Abstände werden ebenfalls aus dem Datenset entfernt.

```
# STRIPWHITESPACE
corp_ds2 = tm_map(corp_ds2, stripWhitespace)
```

```
# REMOVING STOPWORDS
corp_ds2 <- tm_map(corp_ds2, removeWords, stopwords("german"))
```

```
# STEMMING
corp_ds2 <- tm_map(corp_ds2,stemDocument)
#writeLines(as.character(corp_ds2[[3]]))
```

Erstellen von Matrizen

Mit den vorverarbeiteten Texten kann nun eine Matrix erstellt werden. Zunächst wird eine `term-document`-Matrix erstellt und danach eine `document-term`-Matrix. Bei letzterer muss angegeben werden, welche Gewichtungsfunktion verwendet wird, in diesem Fall wird anhand `tf-idf` gewichtet.

```
# term document matrix
tdm_ds2 <- TermDocumentMatrix(corp_ds2, control = list(removeSparseTerms = TRUE,
                                                       removePunctuation = TRUE,
                                                       removeNumbers = TRUE,
                                                       stopwords = TRUE,
                                                       stemming = TRUE))

# document term matrix
dtm_ds2 <- DocumentTermMatrix(corp_ds2,
                              control = list(weighting = function(x)
                                weightTfIdf(x, normalize = FALSE), stopwords = TRUE))
```

```
tdm_ds2
```

```
## <<TermDocumentMatrix (terms: 7904, documents: 3121)>>
## Non-/sparse entries: 123253/24545131
## Sparsity           : 100%
## Maximal term length: 140
## Weighting          : term frequency (tf)
```

Die `document-term`-Matrix ist nur sehr dünn besetzt. Um den Speicherbedarf zu reduzieren, werden einige Terme weggelassen und somit die Dimension reduziert.

```
dtm_ds2
```

```
## <<DocumentTermMatrix (documents: 3121, terms: 9197)>>
## Non-/sparse entries: 128169/28575668
## Sparsity           : 100%
## Maximal term length: 140
## Weighting          : term frequency - inverse document frequency (tf-idf)
```

```
# A term-document matrix where those terms from x are removed
# which have at least a sparse percentage of empty
# (i.e., terms occurring 0 times in a document) elements.
# Resulting matrix contains only terms with a sparse factor less than 0.95
dtm_ds2 <- removeSparseTerms(dtm_ds2, 0.90)
dtm_ds2
```

```
## <<DocumentTermMatrix (documents: 3121, terms: 29)>>
## Non-/sparse entries: 13820/76689
```

```
## Sparsity           : 85%
## Maximal term length: 11
## Weighting          : term frequency - inverse document frequency (tf-idf)
```

```
inspect(tdm_ds2)
```

```
## <<TermDocumentMatrix (terms: 7904, documents: 3121)>>
## Non-/sparse entries: 123253/24545131
## Sparsity           : 100%
## Maximal term length: 140
## Weighting          : term frequency (tf)
## Sample             :
##          Docs
## Terms     1799 1800 1801 1802 1805 1806 1808 1835 1858 2271
##   antrag     6    6    6    6    6    6    6    6    6    6
##   berat      0    0    0    0    0    0    0    0    0    0
##   button     5    5    5    5    5    5    5    5    5    5
##   folgend    3    3    3    3    3    3    3    3    3    3
##   leb        1    1    1    1    1    1    1    1    1    1
##   moglich    2    2    2    2    2    2    2    2    2    2
##   partn      0    0    0    0    0    0    0    0    0    0
##   reit       0    0    0    0    0    0    0    0    0    0
##   tarif      0    0    0    0    0    0    0    0    0    0
##   vorhand    2    2    2    2    2    2    2    2    2    2
```

```
inspect(dtm_ds2)
```

```
## <<DocumentTermMatrix (documents: 3121, terms: 29)>>
## Non-/sparse entries: 13820/76689
## Sparsity           : 85%
## Maximal term length: 11
## Weighting          : term frequency - inverse document frequency (tf-idf)
## Sample             :
##       Terms
## Docs   angezeigt antrag beratung button    leben  möglich  partner   reiter
##   2739         0      0 7.505894      0 33.99607 6.353834 4.840881 27.37737
##   2757         0      0 7.505894      0 33.99607 6.353834 4.840881 27.37737
##   2758         0      0 7.505894      0 33.99607 6.353834 4.840881 27.37737
##   2869         0      0 7.505894      0 33.99607 6.353834 4.840881 27.37737
##   2873         0      0 7.505894      0 33.99607 6.353834 4.840881 27.37737
##   2876         0      0 7.505894      0 33.99607 6.353834 4.840881 27.37737
##   2888         0      0 7.505894      0 31.16306 8.471779 4.840881 29.86622
##   2895         0      0 7.505894      0 31.16306 8.471779 4.840881 29.86622
##   2917         0      0 7.505894      0 31.16306 8.471779 4.840881 29.86622
##   2919         0      0 7.505894      0 31.16306 8.471779 4.840881 29.86622
##       Terms
## Docs      tarif vorhanden
##   2739 46.67906  7.942963
##   2757 46.67906  7.942963
##   2758 46.67906  7.942963
##   2869 46.67906  7.942963
##   2873 46.67906  7.942963
##   2876 46.67906  7.942963
##   2888 37.34325  7.942963
##   2895 37.34325  7.942963
```

```
##   2917 37.34325  7.942963
##   2919 37.34325  7.942963
```

Darstellen einer Wortwolke

Zur Veranschaulichung der reduzierten `document-term`-Matrix wird eine Wortwolke erstellt, in der die wichtigsten Begriffe des Datensets gezeigt werden.

```
freq = data.frame(sort(colSums(as.matrix(dtm_ds2)), decreasing=TRUE))
wordcloud(rownames(freq), freq[,1], max.words=100, colors=brewer.pal(5, "Dark2"))
```

Konvertieren der Dokument Term Matrix zu Dataframe

Um die konstruierte Matrix mit den anderen Prädiktoren zusammengefügt werden kann, muss diese zunächst in einen Dataframe transformiert werden.

```
tmp_df_ds2 <- tidy(dtm_ds2)
head(tmp_df_ds2)
```

```
## # A tibble: 6 x 3
##   document term     count
##   <chr>    <chr>    <dbl>
## 1 1        feld      3.30
## 2 1        folgend   2.29
## 3 1        möglich   4.24
## 4 1        partner   2.42
## 5 1        sichtbar  3.25
## 6 1        sieh      4.65
```

```
tmp_df_ds2 <- tmp_df_ds2 |> pivot_wider(names_from = term, values_from = count,
                                        names_repair = "unique", values_fill = 0)
```

```
colnames(tmp_df_ds2)[1] <- "doc_id"
tmp_df_ds2$doc_id <- as.integer(tmp_df_ds2$doc_id)
tmp_df_ds2$row_sum <- rowSums(tmp_df_ds2)
```

```
rbind(tmp_df_ds2, sum(tmp_df_ds2[, 1:length(tmp_df_ds2)]))
```

```
## # A tibble: 2,778 x 31
##    doc_id  feld folgend möglich partner sichtbar  sieh angezeigt button  reqm
##     <dbl> <dbl>   <dbl>   <dbl>   <dbl>    <dbl> <dbl>     <dbl>  <dbl> <dbl>
##  1      1  3.30    2.29    4.24    2.42     3.25  4.65      0      0     0
##  2      2  3.30    2.29    4.24    2.42     3.25  4.65      0      0     0
##  3      3  0       0       0       0        0     0         2.35   5.46  4.23
##  4      4  0       0       0       0        0     0         2.35   5.46  4.23
##  5      5  0       0       8.47    0        0     0         4.70  10.9   0
##  6      6  0       0       0       0        0     0         0      0     0
##  7      7  3.30    0       2.12    0        3.25  0         0      0     0
##  8      8  0       2.29    0       0        0     0         0      0     0
##  9      9  0       0       2.12    0        0     0         0      0     0
## 10     10  0       0       0       0        0     0         0      0     0
## # ... with 2,768 more rows, and 21 more variables: allgemein <dbl>, bzw <dbl>,
## #   kontroll <dbl>, testfäll <dbl>, beratung <dbl>, reiter <dbl>, antrag <dbl>,
## #   daten <dbl>, wurd <dbl>, vertrag <dbl>, gibt <dbl>, leben <dbl>, kfz <dbl>,
## #   vorhanden <dbl>, beim <dbl>, status <dbl>, automatisch <dbl>,
## #   auswahl <dbl>, felder <dbl>, tarif <dbl>, row_sum <dbl>
```

Zusammenführen mit anderen Prädiktoren

Im Folgenden werden alle Prädiktoren in einem Dataframe zusammengefasst. Dieser stellt die Ausgangslage für die Klassifikation dar.

```
final_ds2 <- cbind(ds2$ANF_RISIKO, ds2$ANF_FEHLERWAHRSCHEINLICHKEIT,
                   ds2$ANF_FEHLERKOSTEN, ds2$TF_ABDECKUNG, ds2$AKT_RES_RELEASE,
                   ds2$AKT_RES_STATUS, tmp_df_ds2[1:3121, 2:28])
```

```
# remove rows with NA and not-needed cols
final_ds2 <- final_ds2[1:2718, -c(2,3)]
```

```
df <- final_ds2
names(df)[names(df)=="ds2$ANF_RISIKO"] <- "ANF_RISIKO"
names(df)[names(df)=="ds2$TF_ABDECKUNG"] <- "TF_ABDECKUNG"
names(df)[names(df)=="ds2$AKT_RES_RELEASE"] <- "AKT_RES_RELEASE"
names(df)[names(df)=="ds2$AKT_RES_STATUS"] <- "AKT_RES_STATUS"
```

```
summary(df)
```

```
##   ANF_RISIKO         TF_ABDECKUNG    AKT_RES_RELEASE    AKT_RES_STATUS
##  Length:2718        Min.   : -0.70   Length:2718        Length:2718
##  Class :character   1st Qu.:  2.78   Class :character   Class :character
##  Mode  :character   Median : 16.70   Mode  :character   Mode  :character
##                     Mean   : 30.99
##                     3rd Qu.: 50.00
##                     Max.   :100.00
##       feld            folgend           möglich           partner
```

```
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.000
##  Mean   : 0.6467   Mean   : 0.8292   Mean   : 0.9865   Mean   : 1.385
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 2.1179   3rd Qu.: 0.000
##  Max.   :26.4321   Max.   :13.7287   Max.   :16.9436   Max.   :67.772
##     sichtbar            sieh           angezeigt           button
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.000
##  Mean   : 0.7652   Mean   : 0.9538   Mean   : 0.9555   Mean   : 1.303
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.000
##  Max.   :42.3104   Max.   :18.6162   Max.   :14.0883   Max.   :35.507
##       reqm           allgemein            bzw             kontroll
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   :0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median :0.0000
##  Mean   : 0.8781   Mean   : 0.5879   Mean   : 0.7533   Mean   :0.4186
##  3rd Qu.: 2.1159   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.:0.0000
##  Max.   :12.6956   Max.   :26.2509   Max.   :17.6761   Max.   :2.9784
##     testfäll         beratung          reiter           antrag
##  Min.   :0.0000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median :0.0000   Median : 0.000   Median : 0.000   Median : 0.0000
##  Mean   :0.4211   Mean   : 1.039   Mean   : 1.454   Mean   : 0.9739
##  3rd Qu.:0.0000   3rd Qu.: 1.876   3rd Qu.: 0.000   3rd Qu.: 0.0000
##  Max.   :2.9496   Max.   :20.641   Max.   :29.866   Max.   :28.0776
##      daten             wurd            vertrag             gibt
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.682   Mean   : 0.6296   Mean   : 0.7823   Mean   : 0.5401
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :15.421   Max.   :23.2035   Max.   :51.8635   Max.   :13.1075
##      leben             kfz            vorhanden           beim
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.0000   Median : 0.000   Median : 0.0000
##  Mean   : 1.343   Mean   : 0.9435   Mean   : 1.133   Mean   : 0.8293
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 1.986   3rd Qu.: 0.0000
##  Max.   :33.996   Max.   :29.1083   Max.   :17.872   Max.   :28.0366
##      status          automatisch         auswahl
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.6898   Mean   : 0.5935   Mean   : 0.6715
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :24.1526   Max.   :19.0892   Max.   :34.9299
```

Normalisieren numerischer Spalten

Mittels `min-max`-Normalisierung werden die numerischen Spalten auf eine gemeinsame Skalierung gebracht. Zur besseren Übersicht wird am Ende nochmal eine Zusammenfassung ausgegeben.

```r
set.seed(1234)
# definiere normalisierungsfunktion
min_max_norm <- function(x) {
(x - min(x)) / (max(x) - min(x))
}
# alle spalten normalisieren
df[, 5:31] <- as.data.frame(lapply(df[, 5:31], 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_RELEASE AKT_RES_STATUS
##  gering: 561   Min.   :0.00000   21x     :1016   FAILED: 469
##  hoch  : 913   1st Qu.:0.03456   22.10   : 346   OK    :2112
##  mittel:1244   Median :0.17279   22.20   : 598   OPEN  : 137
##                Mean   :0.31474   22.30   : 285
##                3rd Qu.:0.50348   OLDERT21: 473
##                Max.   :1.00000
##       feld             folgend          möglich           partner
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.0000   Median :0.00000   Median :0.00000
##  Mean   :0.02447   Mean   :0.0604   Mean   :0.05822   Mean   :0.02043
##  3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.12500   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##     sichtbar            sieh           angezeigt           button
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.01808   Mean   :0.05123   Mean   :0.06782   Mean   :0.03671
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##       reqm           allgemein           bzw            kontroll
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.0000   Median :0.00000   Median :0.0000
##  Mean   :0.06917   Mean   :0.0224   Mean   :0.04262   Mean   :0.1405
##  3rd Qu.:0.16667   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##     testfäll         beratung          reiter           antrag
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.0000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.1428   Mean   :0.05034   Mean   :0.04869   Mean   :0.03468
##  3rd Qu.:0.0000   3rd Qu.:0.09091   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##      daten              wurd            vertrag            gibt
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.04422   Mean   :0.02713   Mean   :0.01508   Mean   :0.04121
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
```

```
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##      leben              kfz            vorhanden           beim
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.00000
##  Mean   :0.03949   Mean   :0.03241   Mean   :0.0634   Mean   :0.02958
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.1111   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
##      status         automatisch         auswahl
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.02856   Mean   :0.03109   Mean   :0.01922
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
```

Klassifikation

Erstellen von Train- / Test-Split

Die vorliegenden Daten werden in Trainings- und Testdaten aufgeteilt im Verhältnis 80:20.

```
# 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

Ein Naive-Bayes Klassifikator wird erstellt und mit den Trainingsdaten trainiert. Anhand der Testdaten wird das Modell evaluiert. Die Ergebnisse werden in einer Confusionmatrix angegeben.

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

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##    gering      hoch    mittel
## 0.2063419 0.3359375 0.4577206
##
## Conditional probabilities:
##         TF_ABDECKUNG
## Y             [,1]      [,2]
##   gering 0.5168818 0.3585291
```

```
##   hoch   0.2369333 0.2801464
##   mittel 0.2890796 0.3314065
##
##         AKT_RES_RELEASE
## Y              21x      22.10      22.20      22.30   OLDERT21
##   gering 0.36525612 0.10022272 0.14922049 0.15590200 0.22939866
##   hoch   0.36662107 0.19151847 0.20246238 0.08071135 0.15868673
##   mittel 0.37048193 0.09738956 0.27510040 0.09236948 0.16465863
##
##         AKT_RES_STATUS
## Y            FAILED         OK       OPEN
##   gering 0.15590200 0.79064588 0.05345212
##   hoch   0.14911081 0.81942544 0.03146375
##   mittel 0.19578313 0.73995984 0.06425703
##
##         feld
## Y              [,1]       [,2]
##   gering 0.03034521 0.09856302
##   hoch   0.02171683 0.06969474
##   mittel 0.02384538 0.08515385
##
##         folgend
## Y              [,1]      [,2]
##   gering 0.07683742 0.1446739
##   hoch   0.05494756 0.1342358
##   mittel 0.05923695 0.1300883
##
##         möglich
## Y              [,1]      [,2]
##   gering 0.05790646 0.1192371
##   hoch   0.05677155 0.1264653
##   mittel 0.06099398 0.1312421
##
##         partner
## Y              [,1]       [,2]
##   gering 0.02577156 0.07747452
##   hoch   0.01920070 0.06145820
##   mittel 0.02047476 0.07311058
##
##         sichtbar
## Y              [,1]       [,2]
##   gering 0.01644680 0.05647460
##   hoch   0.01946754 0.07457863
##   mittel 0.01652765 0.06552692
##
##         sieh
## Y              [,1]      [,2]
##   gering 0.06180401 0.1344420
##   hoch   0.05232558 0.1313382
##   mittel 0.04869478 0.1228363
##
##         angezeigt
## Y              [,1]      [,2]
##   gering 0.07720861 0.1620728
```

```
##   hoch   0.06862745 0.1609829
##   mittel 0.07011379 0.1603261
##
##         button
## Y              [,1]       [,2]
##   gering 0.04283022 0.11577326
##   hoch   0.03819846 0.11043994
##   mittel 0.03320976 0.09093024
##
##         reqm
## Y              [,1]      [,2]
##   gering 0.05716407 0.1334403
##   hoch   0.07204742 0.1591212
##   mittel 0.07262383 0.1559259
##
##         allgemein
## Y              [,1]       [,2]
##   gering 0.02867483 0.08944193
##   hoch   0.02342681 0.07950567
##   mittel 0.01982932 0.07423561
##
##         bzw
## Y              [,1]      [,2]
##   gering 0.03489235 0.1039643
##   hoch   0.04445964 0.1280610
##   mittel 0.04501339 0.1365962
##
##         kontroll
## Y             [,1]      [,2]
##   gering 0.1135857 0.3176614
##   hoch   0.1751026 0.3803151
##   mittel 0.1325301 0.3392365
##
##         testfäll
## Y              [,1]      [,2]
##   gering 0.09799555 0.2976403
##   hoch   0.19015048 0.3926885
##   mittel 0.12751004 0.3337110
##
##         beratung
## Y              [,1]      [,2]
##   gering 0.04677060 0.0990804
##   hoch   0.05621191 0.1233754
##   mittel 0.04937934 0.0969292
##
##         reiter
## Y              [,1]      [,2]
##   gering 0.03656273 0.1255300
##   hoch   0.05380757 0.1621897
##   mittel 0.04426037 0.1332505
##
##         antrag
## Y              [,1]       [,2]
##   gering 0.03219275 0.09552939
```

```
##   hoch   0.03158811 0.09286457
##   mittel 0.03924790 0.11856099
##
##         daten
## Y              [,1]      [,2]
##   gering 0.04944321 0.1437877
##   hoch   0.04952120 0.1373294
##   mittel 0.04016064 0.1258282
##
##         wurd
## Y              [,1]       [,2]
##   gering 0.02672606 0.07024411
##   hoch   0.02582079 0.06982863
##   mittel 0.03012048 0.08370435
##
##         vertrag
## Y              [,1]       [,2]
##   gering 0.01503341 0.06482978
##   hoch   0.01265390 0.06460644
##   mittel 0.01606426 0.05083323
##
##         gibt
## Y              [,1]      [,2]
##   gering 0.04788419 0.1364856
##   hoch   0.04138167 0.1298614
##   mittel 0.04166667 0.1298272
##
##         leben
## Y              [,1]      [,2]
##   gering 0.03062361 0.1189225
##   hoch   0.04331965 0.1539208
##   mittel 0.03681392 0.1360902
##
##         kfz
## Y              [,1]       [,2]
##   gering 0.02427617 0.08769608
##   hoch   0.03160055 0.09637276
##   mittel 0.03423695 0.11074222
##
##         vorhanden
## Y              [,1]      [,2]
##   gering 0.06978471 0.1369796
##   hoch   0.05730354 0.1223100
##   mittel 0.06314145 0.1321912
##
##         beim
## Y              [,1]       [,2]
##   gering 0.03273942 0.10678678
##   hoch   0.02667579 0.08382812
##   mittel 0.03293173 0.09457570
##
##         status
## Y              [,1]       [,2]
##   gering 0.03368597 0.10504702
```

```
##   hoch   0.03556772 0.10517591
##   mittel 0.02183735 0.07216179
##
##         automatisch
## Y              [,1]       [,2]
##   gering 0.03637713 0.11318416
##   hoch   0.02758778 0.08434799
##   mittel 0.03078983 0.09997320
##
##         auswahl
## Y              [,1]       [,2]
##   gering 0.02227171 0.06772879
##   hoch   0.01880985 0.04874339
##   mittel 0.01748661 0.06401465
```

```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     40   32     41
##     hoch       21   58     83
##     mittel     51   92    124
##
## Overall Statistics
##
##                Accuracy : 0.4096
##                  95% CI : (0.3679, 0.4523)
##     No Information Rate : 0.4576
##     P-Value [Acc > NIR] : 0.9890
##
##                   Kappa : 0.0645
##
##  Mcnemar's Test P-Value : 0.2801
##
## Statistics by Class:
##
##                      Class: gering Class: hoch Class: mittel
## Precision                   0.3540      0.3580        0.4644
## Recall                      0.3571      0.3187        0.5000
## F1                          0.3556      0.3372        0.4816
## Prevalence                  0.2066      0.3358        0.4576
## Detection Rate              0.0738      0.1070        0.2288
## Detection Prevalence        0.2085      0.2989        0.4926
## Balanced Accuracy           0.5937      0.5149        0.5068
```

KNN Klassifikation

Analog zum Naive-Bayes Klassifikator wird auch ein KNN Modell trainiert. Auch hier wird das Ergebnis anhand einer Confusionmatrix gezeigt.

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

```
## k-Nearest Neighbors
##
## 2176 samples
##   30 predictor
##    3 classes: 'gering', 'hoch', 'mittel'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1741, 1741, 1740, 1742, 1740
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.4875957  0.1718597
##   7  0.4940452  0.1768661
##   9  0.4935769  0.1744056
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

```r
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     25   10     22
##     hoch       38   88     57
##     mittel     49   84    169
##
## Overall Statistics
##
##                Accuracy : 0.5203
##                  95% CI : (0.4773, 0.5631)
##     No Information Rate : 0.4576
##     P-Value [Acc > NIR] : 0.001972
##
##                   Kappa : 0.2135
##
##  Mcnemar's Test P-Value : 5.848e-07
##
## Statistics by Class:
##
##                      Class: gering Class: hoch Class: mittel
## Precision                  0.43860      0.4809        0.5596
## Recall                     0.22321      0.4835        0.6815
## F1                         0.29586      0.4822        0.6145
## Prevalence                 0.20664      0.3358        0.4576
## Detection Rate             0.04613      0.1624        0.3118
## Detection Prevalence       0.10517      0.3376        0.5572
```

```
## Balanced Accuracy          0.57440      0.6098        0.6145
```