Predictprojcodeincident

suman

May 5, 2018

# Predict project assignment group from the description of incident of ITSM database.

rmarkdown is used and saved as .rmd and then published in kniter

## Motivation

Sometimes incidents are assigned to wrong groups causing a lot of time wasted to reassign to particular group. So the idea was to make it automatically assigned to correct group using predictive modeling and text mining. This has a huge potential to minimise cost and customer satisfaction

## Data collection and import libraries

library(slam)  
library(NLP)  
library(tm)  
library(RColorBrewer)  
library(wordcloud)  
library(e1071)

## Warning: package 'e1071' was built under R version 3.4.1

setwd("C:\\Users\\suman\\Desktop\\datasciencework\\PROJECTS")  
res\_inc=read.csv("res\_inc.csv")  
  
str(res\_inc)

## 'data.frame': 3873 obs. of 3 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ Assignment\_Group: Factor w/ 4 levels "F161 Sys. Mgm. EDW",..: 1 1 1 2 2 3 3 1 1 1 ...  
## $ TITLE : Factor w/ 3790 levels "Change in ZW\_FACT\_KOR\_2XXX\_H view",..: 594 1076 1154 1829 1628 2905 2256 1461 609 1001 ...

table(res\_inc$Assignment\_Group)

##   
## F161 Sys. Mgm. EDW F162 Sys. Mgm. Quality Assurance   
## 1467 706   
## F163 Sys. Mgm. EDW Infrastructure F164 Sys. Mgm. EDW Architecture   
## 1634 66

res\_inc=res\_inc[,-1]

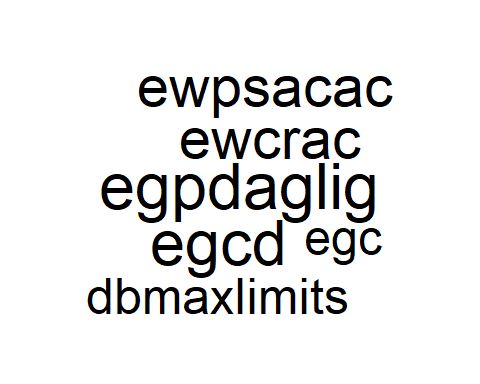
we have 1467,706,1634,66 no of data in each class , total 3873. SO the distribution of data is good compare to production total data

# Data wranglling and wordcloud

res\_inc<-as.data.frame(lapply(res\_inc ,function(x) gsub(".", " ",  
x,fixed = TRUE)))  
#make "." to " "  
  
#create corpora and tdm after cleanning stopwords, punctuation etc  
corp<-Corpus(VectorSource(res\_inc$TITLE))  
inspect(corp[1:3])

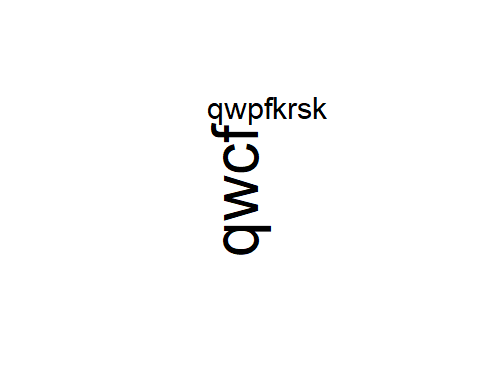
## <<SimpleCorpus>>  
## Metadata: corpus specific: 1, document level (indexed): 0  
## Content: documents: 3  
##   
## [1] JOB EWCFLK08 (EWPCPYLK08 0010 01191800 2624) ENDED IN ERROR 2624 tiludv  
## [2] JOB EWCSHG08 (EWPSAKHG08 0010 01201800 1000) ENDED IN ERROR 1000 tiludv  
## [3] JOB EWCSKS01 (EWPSAKKS01 0010 11131800 1625) ENDED IN ERROR 1625 tiludv

#convert to lowercase  
corpus\_clean <- tm\_map(corp, tolower)  
#remove numbers   
corpus\_clean <- tm\_map(corpus\_clean, removeNumbers)  
#remove english stopwords  
corpus\_clean <- tm\_map(corpus\_clean, removeWords, stopwords())  
#remove punctuations  
corpus\_clean <- tm\_map(corpus\_clean, removePunctuation)  
#unnecessary spaces are removed  
corpus\_clean <- tm\_map(corpus\_clean, stripWhitespace)  
#for specific these group of words some words seems common to all documents removed those  
mystopword<-c("a","an","the","job","error","automatic","batch","tiludv","auto","ended","jcl","jcli")  
corpus\_clean <- tm\_map(corpus\_clean, removeWords,mystopword)  
#create term document matrix with TF   
sms\_dtm <- DocumentTermMatrix(corpus\_clean)  
#sms\_dtm <- DocumentTermMatrix(corpus\_clean,control=list(weighting=weightTfIdf))  
set.seed(123)  
#split train and test  
smp\_size <- floor(0.75 \* nrow(res\_inc))  
train\_ind <- sample(seq\_len(nrow(res\_inc)), size = smp\_size)  
sms\_raw\_train <- res\_inc[train\_ind, ]  
sms\_raw\_test <- res\_inc[-train\_ind, ]  
sms\_dtm\_train <- sms\_dtm[train\_ind, ]  
sms\_dtm\_test <- sms\_dtm[-train\_ind, ]  
sms\_corpus\_train <- corpus\_clean[train\_ind]  
sms\_corpus\_test <- corpus\_clean[-train\_ind]  
  
  
ind<-which(sms\_raw\_train$Assignment\_Group=="F161 Sys Mgm EDW")  
sms\_corpus\_train\_f161 <- sms\_corpus\_train[ind]  
ind<-which(sms\_raw\_train$Assignment\_Group=="F162 Sys Mgm Quality Assurance")  
sms\_corpus\_train\_f162 <- sms\_corpus\_train[ind]  
ind<-which(sms\_raw\_train$Assignment\_Group=="F163 Sys Mgm EDW Infrastructure")  
sms\_corpus\_train\_f163 <- sms\_corpus\_train[ind]  
ind<-which(sms\_raw\_train$Assignment\_Group=="F164 Sys Mgm EDW Architecture")  
sms\_corpus\_train\_f164 <- sms\_corpus\_train[ind]  
#get wordcloud for diff proj code  
  
wordcloud(sms\_corpus\_train\_f161, min.freq = 40, random.order = FALSE)



wordcloud(sms\_corpus\_train\_f162, min.freq = 30, random.order = FALSE)

## Warning in wordcloud(sms\_corpus\_train\_f162, min.freq = 30, random.order =  
## FALSE): qwpendtoend could not be fit on page. It will not be plotted.



wordcloud(sms\_corpus\_train\_f163, min.freq = 50, random.order = FALSE)

## Warning in wordcloud(sms\_corpus\_train\_f163, min.freq = 50, random.order =  
## FALSE): zwcdgfaczpa could not be fit on page. It will not be plotted.

## Warning in wordcloud(sms\_corpus\_train\_f163, min.freq = 50, random.order =  
## FALSE): zwcdgfakorzpa could not be fit on page. It will not be plotted.

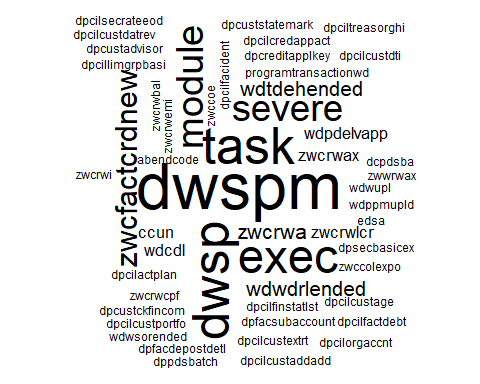
## Warning in wordcloud(sms\_corpus\_train\_f163, min.freq = 50, random.order =  
## FALSE): zwcdgkundezpa could not be fit on page. It will not be plotted.



wordcloud(sms\_corpus\_train\_f164, min.freq = 50, random.order = FALSE)

## Warning in wordcloud(sms\_corpus\_train\_f164, min.freq = 50, random.order =  
## FALSE): dpcredapprover could not be fit on page. It will not be plotted.

## Warning in wordcloud(sms\_corpus\_train\_f164, min.freq = 50, random.order =  
## FALSE): zwpadhocnoncm could not be fit on page. It will not be plotted.

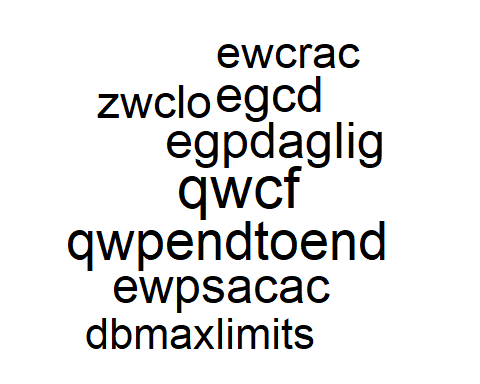


wordcloud(sms\_corpus\_train, min.freq = 50, random.order = FALSE)

## Warning in wordcloud(sms\_corpus\_train, min.freq = 50, random.order =  
## FALSE): zwcdgfaczpa could not be fit on page. It will not be plotted.

## Warning in wordcloud(sms\_corpus\_train, min.freq = 50, random.order =  
## FALSE): zwcdgfakorzpa could not be fit on page. It will not be plotted.

## Warning in wordcloud(sms\_corpus\_train, min.freq = 50, random.order =  
## FALSE): zwcdgkundezpa could not be fit on page. It will not be plotted.

 # Split data into train and test

sms\_train <- DocumentTermMatrix(sms\_corpus\_train)  
sms\_test <- DocumentTermMatrix(sms\_corpus\_test)  
sms\_train.m <- as.matrix(sms\_train)  
sms\_test.m <- as.matrix(sms\_test)  
sms\_train.df <- as.data.frame(sms\_train.m)  
sms\_test.df <- as.data.frame(sms\_test.m)  
  
#naive bayes only works with catagorigal data but DTM contains count  
#so we will change it to 1 or 0  
  
convert\_counts <- function(x) {  
 x <- ifelse(x > 0, 1, 0)  
 x <- factor(x, levels = c(0, 1), labels = c("No", "Yes"))  
 return(x)  
}  
sms\_train.df <- apply(sms\_train.df, MARGIN = 2, convert\_counts)  
sms\_test.df <- apply(sms\_test.df, MARGIN = 2, convert\_counts)

# Build model naive bayes

sms\_classifier <- naiveBayes(sms\_train.df, as.factor(sms\_raw\_train$Assignment\_Group),laplace=1)  
sms\_test\_pred\_e <- predict(sms\_classifier, sms\_test.df)

# Performance measurement of model

Naive bayes model is giving 93% accuracy on test data

a<-table(True=sms\_raw\_test$Assignment\_Group,Pred=sms\_test\_pred\_e)  
  
  
I=dim(a)[1]  
J=dim(a)[2]  
true<-0  
false<-0  
for (i in 1:I){  
 for (j in 1:J){  
 if (i==j){true=true+a[i,j]}}}  
for (i in 1:I){  
 for (j in 1:J){  
 if (i!=j){  
 false=false+a[i,j]}}}  
accuracy=true/(false+true)  
print(accuracy)

## [1] 0.9329205

# Predict single case

input<-"JOB EWCSCR41 (EWPSAKRW14.0010.01041800.JCL) ENDED IN ERROR JCL tiludv"  
input<-gsub(".", " ", input,fixed = TRUE)  
corp1<-Corpus(VectorSource(input))  
corpus\_clean1 <- tm\_map(corp1, tolower)  
corpus\_clean1 <- tm\_map(corpus\_clean1, removeNumbers)  
corpus\_clean1 <- tm\_map(corpus\_clean1, removeWords, stopwords())  
corpus\_clean1 <- tm\_map(corpus\_clean1, removePunctuation)  
corpus\_clean1 <- tm\_map(corpus\_clean1, stripWhitespace)  
corpus\_clean1 <- tm\_map(corpus\_clean1, removeWords,mystopword)  
sms\_corpus\_test1 <- corpus\_clean1  
sms\_test1 <- DocumentTermMatrix(sms\_corpus\_test1)  
sms\_test1.m <- as.matrix(sms\_test1)  
sms\_test1.df <- as.data.frame(sms\_test1.m)  
sms\_test1.df <- apply(sms\_test1.df, MARGIN = 2, convert\_counts)  
sms\_test\_pred <- predict(sms\_classifier, sms\_test1.df)  
print(sms\_test\_pred)

## [1] F163 Sys Mgm EDW Infrastructure F163 Sys Mgm EDW Infrastructure  
## 4 Levels: F161 Sys Mgm EDW ... F164 Sys Mgm EDW Architecture

# Try a different model SVM so that we can choose best one

SVm cannot work with factor(yes/no) so removed convert\_counts methood

# Data cleanning same way like above and create TDM

res\_inc$Assignment\_Group<-as.character(res\_inc$Assignment\_Group)  
res\_inc$TITLE<-as.character(res\_inc$TITLE)  
res\_inc$TITLE<-tolower(res\_inc$TITLE)  
res\_inc$Assignment\_Group<-as.factor(res\_inc$Assignment\_Group)  
res\_inc$TITLE<-as.factor(res\_inc$TITLE)  
#1073 has junk char as danish so that should be removed else those will create some junk char and wont match in test  
to.plain <- function(s) {  
  
 # 1 character substitutions  
 old1 <- "åÅøä"  
 new1 <- "aaoa"  
 s1 <- chartr(old1, new1, s)  
 # 2 character substitutions  
 old2 <- c("o", "ß", "æ", "ø")  
 new2 <- c("oe", "ss", "ae", "oe")  
 s2 <- s1  
 for(i in seq\_along(old2)) s2 <- gsub(old2[i], new2[i], s2, fixed = TRUE)  
  
 s2  
}  
subset(res\_inc, grepl("ew\_wh", TITLE))

## [1] Assignment\_Group TITLE   
## <0 rows> (or 0-length row.names)

res\_inc<-as.data.frame(lapply(res\_inc ,to.plain))  
res\_inc<-as.data.frame(lapply(res\_inc ,function(x) gsub(".", " ",  
x,fixed = TRUE)))  
corp<-Corpus(VectorSource(res\_inc$TITLE))  
inspect(corp[1:3])

## <<SimpleCorpus>>  
## Metadata: corpus specific: 1, document level (indexed): 0  
## Content: documents: 3  
##   
## [1] joeb ewcflk08 (ewpcpylk08 0010 01191800 2624) ended in erroer 2624 tiludv  
## [2] joeb ewcshg08 (ewpsakhg08 0010 01201800 1000) ended in erroer 1000 tiludv  
## [3] joeb ewcsks01 (ewpsakks01 0010 11131800 1625) ended in erroer 1625 tiludv

corpus\_clean <- tm\_map(corp, tolower)  
corpus\_clean <- tm\_map(corpus\_clean, removeNumbers)  
corpus\_clean <- tm\_map(corpus\_clean, removeWords, stopwords())  
corpus\_clean <- tm\_map(corpus\_clean, removePunctuation)  
corpus\_clean <- tm\_map(corpus\_clean, stripWhitespace)  
mystopword<-c("a","an","the","job","error","automatic","batch","tiludv","auto","ended","jcl","jcli")  
corpus\_clean <- tm\_map(corpus\_clean, removeWords,mystopword)  
set.seed(123)  
sms\_all <- DocumentTermMatrix(corpus\_clean)  
sms\_all.m <- as.matrix(sms\_all)  
sms\_all.df <- as.data.frame(sms\_all.m)  
#sort columns in ascending so that train and test columns are in same order, also same terms are used in train and test else svm wont work  
cc<-colnames(sms\_all.df)[order(colnames(sms\_all.df))]  
sms\_all.df<-sms\_all.df[,cc]  
all<-as.factor(res\_inc$Assignment\_Group)  
data <- as.data.frame(cbind(all,as.matrix(sms\_all.df)))  
smp\_size <- floor(0.75 \* nrow(data))  
train\_ind <- sample(seq\_len(nrow(data)), size = smp\_size)  
data\_train<-data[train\_ind,]#2859 1325 col and 1 predictor  
data\_test<-data[-train\_ind,]#953  
sv <- svm(all~., data\_train, type="C-classification", kernel="linear", cost=1)

## Warning in svm.default(x, y, scale = scale, ..., na.action = na.action):  
## Variable(s) 'bylkdayudnyt' and 'byswaph' and 'client' and 'dccdpm'  
## and 'dcimftp' and 'dmystart' and 'dpcilcredit' and 'dpcilfacaccg' and  
## 'dpcustcoerpcoed' and 'dpdplreaimp' and 'dpgfsdlcrdepoes' and 'ducloeib'  
## and 'ducloeibzpa' and 'ducloeiz' and 'ducloeizzpa' and 'ducnspd' and  
## 'ducnspdzpa' and 'ducnspe' and 'ducnspezpa' and 'ducnspi' and 'ducnspizpa'  
## and 'ducnspn' and 'ducnspnzpa' and 'ducnsps' and 'ducnspszpa' and  
## 'ducunls' and 'ducunlszpa' and 'etctih' and 'etpkapdwhtfr' and 'ewbsunl'  
## and 'ewcfks' and 'ewcflk' and 'ewcfzy' and 'ewcroeftp' and 'ewcsbr'  
## and 'ewcskp' and 'ewcsll' and 'ewcsnb' and 'ewcsu' and 'ewcswf' and  
## 'ewcxcm' and 'ewpcpyf' and 'ewpcpyfa' and 'ewpcpyks' and 'ewpcpylk'  
## and 'ewpcpyzy' and 'ewpjoeurnaloepryd' and 'ewpsakbr' and 'ewpsakdd'  
## and 'ewpsakdm' and 'ewpsakkp' and 'ewpsaklkny' and 'ewpsaknb' and  
## 'ewpsaksk' and 'ewpsaku' and 'incident' and 'incoerrect' and 'install'  
## and 'ktoekrsinren' and 'ktoekrsrente' and 'lkprwadaakts' and 'lkprwadam'  
## and 'lkprwakoencnsoelv' and 'march' and 'name' and 'oecbzctstzpa' and  
## 'oecwzctst' and 'perfoermance' and 'qwcaks' and 'qwcdcraa' and 'qwcdegh'  
## and 'qwcdfpoec' and 'qwcdkoemt' and 'qwcdlgmibvap' and 'qwcoenk' and  
## 'qwcqdd' and 'qwcqdw' and 'qwcqex' and 'qwcqhb' and 'qwcqoei' and 'qwcqu'  
## and 'qwcrsi' and 'qwcvfppm' and 'qwcvgmi' and 'qwcvks' and 'qwdcpftpppm'  
## and 'qwpcilgfsbsumeoed' and 'qwpdcraazpa' and 'qwpdkoemt' and 'qwpeldcamdg'  
## and 'qwpkkszpa' and 'qwwclcra' and 'qwwed' and 'qwwtac' and 'qwwtfa'  
## and 'relea' and 'releas' and 'reprices' and 'sent' and 'test' and 'text'  
## and 'toeoels' and 'wdciv' and 'wdpintvalapp' and 'xrsrms' and 'zwbfi'  
## and 'zwcelltv' and 'zwcesdr' and 'zwceusa' and 'zwceve' and 'zwcevmpg'  
## and 'zwcevsi' and 'zwcevys' and 'zwcexk' and 'zwcexy' and 'zwcfgdf' and  
## 'zwcfkkr' and 'zwcfknb' and 'zwcfksu' and 'zwcfoek' and 'zwcfpc' and  
## 'zwcfplfr' and 'zwcfvsa' and 'zwcgms' and 'zwckaftb' and 'zwckgunl' and  
## 'zwckmcl' and 'zwckmmgr' and 'zwckoem' and 'zwckxyf' and 'zwclgd' and  
## 'zwclgx' and 'zwcltv' and 'zwcnkra' and 'zwcoedm' and 'zwcoeroe' and  
## 'zwcrwalx' and 'zwcrwys' and 'zwcscyk' and 'zwctbbbm' and 'zwcudap' and  
## 'zwculfactpoezpa' and 'zwculfakproeactrl' and 'zwculfaksiloeswe' and  
## 'zwcwwp' and 'zwcwwycl' and 'zwcwwys' and 'zwkundidh' and 'zwpdgkaft' and  
## 'zwpdglgddmb' and 'zwprwalgdregpd' and 'zwpuldefzpa' and 'zwpulkredfactexp'  
## constant. Cannot scale data.

a<-table(Pred=predict(sv, data\_test[,-1]) , True=data\_test$all) #1325  
I=dim(a)[1]  
J=dim(a)[2]  
true<-0  
false<-0  
for (i in 1:I){  
 for (j in 1:J){  
 if (i==j){true=true+a[i,j]}  
 }  
}  
  
for (i in 1:I){  
 for (j in 1:J){  
 if (i!=j){  
 false=false+a[i,j]}  
 }  
}

# check accuracy of SVM which is also giving 93% accuracy

accuracy=true/(false+true)  
print(accuracy)

## [1] 0.9432405

# predict single case using SVM model

#input<-"JOB EWCRRW14 (EWPSAKRW14.0010.01041800.JCL) ENDED IN ERROR JCL tiludv"  
input<-"JOB ZWCLEGB3 (ZWCFAC.0010.01041800.JCL) ENDED IN ERROR JCL tiludv suman"  
input<-gsub(".", " ", input,fixed = TRUE)  
corp1<-Corpus(VectorSource(input))  
corpus\_clean1 <- tm\_map(corp1, tolower)  
corpus\_clean1 <- tm\_map(corpus\_clean1, removeNumbers)  
corpus\_clean1 <- tm\_map(corpus\_clean1, removeWords, stopwords())  
corpus\_clean1 <- tm\_map(corpus\_clean1, removePunctuation)  
corpus\_clean1 <- tm\_map(corpus\_clean1, stripWhitespace)  
corpus\_clean1 <- tm\_map(corpus\_clean1, removeWords,mystopword)  
sms\_corpus\_test1 <- corpus\_clean1  
#use the same term as training otherwise for single prediction svm wont work  
sms\_test1 <- DocumentTermMatrix(sms\_corpus\_test1,control =  
list(dictionary=Terms(sms\_all)) )  
sms\_test1.m <- as.matrix(sms\_test1)  
sms\_test1.df <- as.data.frame(sms\_test1.m)  
#sort columns in ascending to get right order as training  
sms\_test1.df<-sms\_test1.df[,cc]  
sms\_test1.df<-as.data.frame(as.matrix(sms\_test1.df))  
a<-predict(sv, sms\_test1.df)#1324 #object 'ewpmåned' not found (butewpmÃ¥ned found)  
print(a)

## 1   
## 3   
## Levels: 1 2 3 4

#colnames(sms\_test1.df)[order(colnames(sms\_test1.df))],  
#because this column is not present in test