Stratified Sample

Mirza Naseh Ahmad

4/25/2021

datacomplete<-as.data.frame(fread('amazon\_reviews\_us\_Automotive\_v1\_00.tsv'),stringsAsFactors = FALSE)

## Warning in fread("amazon\_reviews\_us\_Automotive\_v1\_00.tsv"): Found and  
## resolved improper quoting out-of-sample. First healed line 4209: <<US 35675398  
## R5YPUKJBX85J0 B005G82C8E 826586699 "THE E Z SLIDE SHADE" for Retractable  
## Sunshades Available for All Car Models. Please send the model and car year.  
## Automotive 5 5 6 N Y Best choice on the market. This is so convenient!!! I live  
## in phoenix AZ where the internal temperature of cars can bake cookies. literally  
## look it up on YouTube. I hate fussing with all the sun shades putting them  
## behind the seat where half the time you just don't use them. I looked at other  
## styles of permanent sun >>. If the fields are not quoted (e.g. field separator  
## does not appear within any field), try quote="" to avoid this warning.

summary(datacomplete)

## marketplace customer\_id review\_id product\_id   
## Length:3514942 Min. : 10004 Length:3514942 Length:3514942   
## Class :character 1st Qu.:13048596 Class :character Class :character   
## Mode :character Median :24033489 Mode :character Mode :character   
## Mean :25958150   
## 3rd Qu.:40529903   
## Max. :53096570   
## product\_parent product\_title product\_category star\_rating   
## Min. : 771 Length:3514942 Length:3514942 Min. :1.000   
## 1st Qu.:249665847 Class :character Class :character 1st Qu.:4.000   
## Median :501137686 Mode :character Mode :character Median :5.000   
## Mean :500572179 Mean :4.246   
## 3rd Qu.:752072414 3rd Qu.:5.000   
## Max. :999998981 Max. :5.000   
## helpful\_votes total\_votes vine verified\_purchase   
## Min. : 0.00 Min. : 0.000 Length:3514942 Length:3514942   
## 1st Qu.: 0.00 1st Qu.: 0.000 Class :character Class :character   
## Median : 0.00 Median : 0.000 Mode :character Mode :character   
## Mean : 1.04 Mean : 1.328   
## 3rd Qu.: 1.00 3rd Qu.: 1.000   
## Max. :26132.00 Max. :26382.000   
## review\_headline review\_body review\_date   
## Length:3514942 Length:3514942 Min. :1999-10-24   
## Class :character Class :character 1st Qu.:2013-10-03   
## Mode :character Mode :character Median :2014-09-16   
## Mean :2014-04-28   
## 3rd Qu.:2015-03-26   
## Max. :2015-08-31

datacomplete$reviewlength <- nchar(datacomplete$review\_body)  
  
#Highest rated products.  
top\_rated\_products <- datacomplete %>%  
 group\_by(product\_id) %>%   
 summarize(count\_votes = n()) %>%   
 arrange(desc(count\_votes))  
  
top\_rated\_products1 <- top\_rated\_products[top\_rated\_products$count\_votes > 15 ,]  
summary(top\_rated\_products1)

## product\_id count\_votes   
## Length:35977 Min. : 16.00   
## Class :character 1st Qu.: 20.00   
## Mode :character Median : 29.00   
## Mean : 51.59   
## 3rd Qu.: 50.00   
## Max. :4894.00

head(top\_rated\_products1)

## # A tibble: 6 x 2  
## product\_id count\_votes  
## <chr> <int>  
## 1 B005NLQAHS 4894  
## 2 B000CITK8S 4422  
## 3 B001LHVOVK 3694  
## 4 B00068XCQU 2688  
## 5 B001AIZ5HY 2483  
## 6 B00080QHMM 2069

data<- datacomplete[datacomplete$product\_id %in% top\_rated\_products1$product\_id ,]  
  
  
#Data Prep  
data<-na.omit(data)  
  
#removing zero text values   
zerotext<-data[data$reviewlength == 0 ,]  
head(zerotext)

## marketplace customer\_id review\_id product\_id product\_parent  
## 8434 US 106658 R2YRPPVXX4FKDA B00IZNZOYQ 912240060  
## 9245 US 9070651 R1QEOB3F7KK3YC B00JVUFDOS 202897838  
## 11956 US 3101982 R3LSPIC5GOS1VY B000SOM9GQ 681787945  
## 12285 US 131918 R4TJBG8RSOBAQ B008QPTV2E 874881828  
## 12434 US 33866189 R176S6P5M6G8OX B00SH33WJ8 989335268  
## 13864 US 5261620 R3DG4KH9TS2LTC B00FMOJZHS 555735953  
## product\_title  
## 8434 ABN 4 Cylinder Toyota/Lexus Oil Filter Wrench for Prius, Prius V, Corolla, Matrix, Scion, Lexus 15620-31060 15620-36020  
## 9245 Rough Country Suspension 70507 Curved LED Light Windshield Mount  
## 11956 Bell 22-1-45915-8 Anti-Theft License Plate Fastener  
## 12285 HELLA SLOW Decal SLOW AS FCK JDM Euro Funny Car Window Bumper Vinyl Sticker (package Come with hand decal) stickerciti brand  
## 12434 HOT SYSTEM 12V 3156 3157 3757 4157 54-SMD LED Light bulbs For Car Tail Light Backup Light Turn signal light 2-pack  
## 13864 Hi-Lift HM-800 Hood Mount for Jeep JK  
## product\_category star\_rating helpful\_votes total\_votes vine  
## 8434 Automotive 1 2 3 N  
## 9245 Automotive 5 0 0 N  
## 11956 Automotive 1 11 12 N  
## 12285 Automotive 5 1 1 N  
## 12434 Automotive 1 5 5 N  
## 13864 Automotive 5 0 0 N  
## verified\_purchase review\_headline review\_body review\_date reviewlength  
## 8434 Y Useless tool 2015-08-30 0  
## 9245 Y Five Stars 2015-08-30 0  
## 11956 Y One Star 2015-08-29 0  
## 12285 Y Five Stars 2015-08-29 0  
## 12434 Y One Star 2015-08-29 0  
## 13864 Y Five Stars 2015-08-29 0

data<-data[data$reviewlength != 0,]  
  
  
#filtering out non verified purchases  
vpcount = table(data$verified\_purchase)  
vpcount = as.data.frame(vpcount)  
names(vpcount)[1] = 'Verified purchase'  
head(vpcount)

## Verified purchase Freq  
## 1 N 147791  
## 2 Y 1708112

datavp<-data[data$verified\_purchase != 'N' & data$product\_id %in% top\_rated\_products1$product\_id,]  
datavp$star\_rating <- ordered(datavp$star\_rating, levels = c("5", "4", "3", "2", "1"))  
  
  
table(datavp$verified\_purchase)

##   
## Y   
## 1708112

summary(datavp$reviewlength)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.0 55.0 131.0 213.2 251.0 27177.0

glimpse(datavp)

## Rows: 1,708,112  
## Columns: 16  
## $ marketplace <chr> "US", "US", "US", "US", "US", "US", "US", "US", "US"~  
## $ customer\_id <int> 42462164, 52570308, 184627, 40946484, 35335277, 2583~  
## $ review\_id <chr> "R3NORADVJO6IE6", "R2DA9DOT03UW6I", "R1DB5DA7CWWTI8"~  
## $ product\_id <chr> "B000C7S0TO", "B000GKD5NI", "B0002JMAKW", "B000C5CEK~  
## $ product\_parent <int> 907684644, 105401756, 267002949, 389524802, 81681544~  
## $ product\_title <chr> "Spectra Premium CU1909 Complete Radiator for Toyota~  
## $ product\_category <chr> "Automotive", "Automotive", "Automotive", "Automotiv~  
## $ star\_rating <ord> 5, 5, 5, 5, 4, 5, 5, 5, 3, 1, 5, 1, 5, 5, 5, 5, 5, 4~  
## $ helpful\_votes <int> 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2~  
## $ total\_votes <int> 0, 3, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 2~  
## $ vine <chr> "N", "N", "N", "N", "N", "N", "N", "N", "N", "N", "N~  
## $ verified\_purchase <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y~  
## $ review\_headline <chr> "Five Stars", "Good for the price. Fits fairly good ~  
## $ review\_body <chr> "Put it in fine, no problems. Shipping was decent 5 ~  
## $ review\_date <date> 2015-08-31, 2015-08-31, 2015-08-31, 2015-08-31, 201~  
## $ reviewlength <int> 56, 691, 41, 161, 431, 96, 132, 156, 35, 294, 47, 90~

#Stratified Sampling  
set.seed(1000)  
options(stringsASFacgtors = FALSE)  
sr1<- filter(datavp, star\_rating == 1)  
sr2<- filter(datavp, star\_rating == 2)  
sr3<- filter(datavp, star\_rating == 3)  
sr4<- filter(datavp, star\_rating == 4)  
sr5<- filter(datavp, star\_rating == 5)  
  
sampledata1<- sample\_n(sr1,1000 , replace = FALSE)  
sampledata2<- sample\_n(sr2,1000 , replace = FALSE)  
sampledata3<- sample\_n(sr3,1000 , replace = FALSE)  
sampledata4<- sample\_n(sr4,1000 , replace = FALSE)  
sampledata5<- sample\_n(sr5,1000 , replace = FALSE)  
  
sampledata <- rbind(sampledata1, sampledata2, sampledata3, sampledata4, sampledata5)  
sampledata <- data.table(rating = sampledata$star\_rating ,review = sampledata$review\_body , reviewlength = sampledata$reviewlength)  
  
  
set.seed(1002)  
indexes<- createDataPartition(sampledata$rating, times = 1 ,p = 0.7, list = FALSE)  
train<-sampledata[indexes,]  
test <- sampledata[-indexes,]

#Preprocessing Pipeline  
#1. Tokenize  
#2. lower casing  
#3. stop word removal  
#4. Stemming  
#5. Adding Bigrams  
#6. Transform to DFM  
#7. Ensure Test and train DFM have the same features  
  
#tokenization and cleaning  
  
  
train.tokens <- tokens(train$review,what = "word", remove\_numbers = TRUE, remove\_punct = TRUE, split\_hyphens = TRUE, remove\_symbols = TRUE)  
  
  
train.tokens <- tokens\_tolower(train.tokens)  
  
train.tokens[[105]]

## [1] "the" "part" "about" "supporting"   
## [5] "most" "obd1" "cars" "is"   
## [9] "a" "flat" "out" "lie"   
## [13] "i" "tested" "it" "on"   
## [17] "several" "gm" "obd1" "cars"   
## [21] "and" "none" "worked" "accurately"   
## [25] "there" "is" "no" "data"   
## [29] "error" "checking" "so" "you"   
## [33] "can" "get" "false" "different"   
## [37] "data" "just" "by" "selecting"   
## [41] "different" "vehicles" "on" "the"   
## [45] "same" "car" "i" "had"   
## [49] "different" "sets" "of" "results"   
## [53] "on" "a" "firebird" "v8"   
## [57] "and" "none" "were" "accurate"   
## [61] "when" "compared" "with" "a"   
## [65] "real" "gm" "tech2" "scan"   
## [69] "tool" "selecting" "the" "correct"   
## [73] "vehicle" "did" "not" "even"   
## [77] "work" "it" "reported" "invalid"   
## [81] "setting" "a" "very" "detailed"   
## [85] "investigation" "revealed" "that" "the"   
## [89] "tool" "was" "using" "gm"   
## [93] "truck" "specs" "for" "the"   
## [97] "firebird" "br" "br" "obd2"   
## [101] "performance" "was" "correct" "but"   
## [105] "the" "features" "are" "very"   
## [109] "limited" "you" "can" "get"   
## [113] "the" "same" "features" "elsewhere"   
## [117] "for" "less" "than" "half"   
## [121] "the" "cost" "of" "this"   
## [125] "scan" "tool" "real" "time"   
## [129] "sensor" "status" "data" "not"   
## [133] "include" "in" "this" "tool"   
## [137] "for" "example" "can" "be"   
## [141] "bought" "for" "about" "from"   
## [145] "other" "manufacturers" "br" "br"   
## [149] "i" "returned" "it" "promptly"

train.tokens<- tokens\_select(train.tokens, stopwords(), selection = "remove")  
  
train.tokens[[105]]

## [1] "part" "supporting" "obd1" "cars"   
## [5] "flat" "lie" "tested" "several"   
## [9] "gm" "obd1" "cars" "none"   
## [13] "worked" "accurately" "data" "error"   
## [17] "checking" "can" "get" "false"   
## [21] "different" "data" "just" "selecting"   
## [25] "different" "vehicles" "car" "different"   
## [29] "sets" "results" "firebird" "v8"   
## [33] "none" "accurate" "compared" "real"   
## [37] "gm" "tech2" "scan" "tool"   
## [41] "selecting" "correct" "vehicle" "even"   
## [45] "work" "reported" "invalid" "setting"   
## [49] "detailed" "investigation" "revealed" "tool"   
## [53] "using" "gm" "truck" "specs"   
## [57] "firebird" "br" "br" "obd2"   
## [61] "performance" "correct" "features" "limited"   
## [65] "can" "get" "features" "elsewhere"   
## [69] "less" "half" "cost" "scan"   
## [73] "tool" "real" "time" "sensor"   
## [77] "status" "data" "include" "tool"   
## [81] "example" "can" "bought" "manufacturers"  
## [85] "br" "br" "returned" "promptly"

train.tokens<- tokens\_wordstem(train.tokens, language = "english")  
  
train.tokens[[105]]

## [1] "part" "support" "obd1" "car" "flat"   
## [6] "lie" "test" "sever" "gm" "obd1"   
## [11] "car" "none" "work" "accur" "data"   
## [16] "error" "check" "can" "get" "fals"   
## [21] "differ" "data" "just" "select" "differ"   
## [26] "vehicl" "car" "differ" "set" "result"   
## [31] "firebird" "v8" "none" "accur" "compar"   
## [36] "real" "gm" "tech2" "scan" "tool"   
## [41] "select" "correct" "vehicl" "even" "work"   
## [46] "report" "invalid" "set" "detail" "investig"   
## [51] "reveal" "tool" "use" "gm" "truck"   
## [56] "spec" "firebird" "br" "br" "obd2"   
## [61] "perform" "correct" "featur" "limit" "can"   
## [66] "get" "featur" "elsewher" "less" "half"   
## [71] "cost" "scan" "tool" "real" "time"   
## [76] "sensor" "status" "data" "includ" "tool"   
## [81] "exampl" "can" "bought" "manufactur" "br"   
## [86] "br" "return" "prompt"

#bag of words  
  
train.tokens.dfm <-dfm(train.tokens, tolower = FALSE)  
  
train.tokens.matrix <- as.matrix(train.tokens.dfm)  
  
view(train.tokens.matrix[1:10, 1:100])  
  
dim(train.tokens.matrix)

## [1] 3500 6391

colnames(train.tokens.matrix)[1:25]

## [1] "great" "batteri" "long" "buy" "author"   
## [6] "dealer" "amazon" "void" "warranti" "one"   
## [11] "went" "bad" "just" "year" "free"   
## [16] "replac" "unusu" "optima" "manufactur" "honor"   
## [21] "distributor" "show" "link" "advert" "contact"

train.tokens.dfm

## Document-feature matrix of: 3,500 documents, 6,391 features (99.7% sparse).  
## features  
## docs great batteri long buy author dealer amazon void warranti one  
## text1 1 2 1 1 2 1 4 1 4 1  
## text2 0 0 0 0 0 0 0 0 0 0  
## text3 0 0 0 0 0 0 0 0 0 1  
## text4 0 0 0 0 0 0 0 0 0 0  
## text5 0 0 0 0 0 0 0 0 0 0  
## text6 0 0 0 0 0 0 0 0 0 0  
## [ reached max\_ndoc ... 3,494 more documents, reached max\_nfeat ... 6,381 more features ]

#Cross Validation  
train.tokens.df <-cbind(rating = train$rating, convert(train.tokens.dfm, to = "data.frame"))  
  
#clean column names.   
names(train.tokens.df) <- make.names(names(train.tokens.df))  
# drops <- c("document")  
# train.tokens.df <- train.tokens.df[ , !(names(train.tokens.df) %in% drops)]  
  
# use caret to create stratified(because the data is not balanced) folds for 10-fold cross validation repeated 3 times  
set.seed(33445)  
cv.folds<-createMultiFolds(train$rating, k = 10, times = 3)  
  
cv.cntrl<- trainControl(method = "repeatedcv", number = 10, repeats = 3, index = cv.folds)  
  
#timing the code execution  
start.time <- Sys.time()  
  
#make a cluster to work on 8 logical cores  
cl<-makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
  
 drops <- c("document")  
 train.tokens.df <- train.tokens.df[ , !(names(train.tokens.df) %in% drops)]  
  
rpart.cv.1 <- train(rating ~ ., data = train.tokens.df, method = "rpart", trControl = cv.cntrl, tuneLength = 7)  
svmLinear3.cv.1<- train(rating ~., data = train.tokens.df, method = "svmLinear3", trControl = cv.cntrl, tuneLength = 7)  
  
  
stopCluster(cl)  
  
#Execution time  
total.time<- Sys.time() - start.time  
total.time

## Time difference of 13.86043 mins

svmLinear3.cv.1

## L2 Regularized Support Vector Machine (dual) with Linear Kernel   
##   
## 3500 samples  
## 6391 predictors  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## cost Loss Accuracy Kappa   
## 0.25 L1 0.3861905 0.2327381  
## 0.25 L2 0.3845714 0.2307143  
## 0.50 L1 0.3786667 0.2233333  
## 0.50 L2 0.3780952 0.2226190  
## 1.00 L1 0.3735238 0.2169048  
## 1.00 L2 0.3752381 0.2190476  
## 2.00 L1 0.3704762 0.2130952  
## 2.00 L2 0.3715238 0.2144048  
## 4.00 L1 0.3704762 0.2130952  
## 4.00 L2 0.3707619 0.2134524  
## 8.00 L1 0.3705714 0.2132143  
## 8.00 L2 0.3709524 0.2136905  
## 16.00 L1 0.3704762 0.2130952  
## 16.00 L2 0.3719048 0.2148810  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were cost = 0.25 and Loss = L1.

rpart.cv.1

## CART   
##   
## 3500 samples  
## 6391 predictors  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.006071429 0.3176190 0.14702381  
## 0.007678571 0.3007619 0.12595238  
## 0.007857143 0.2956190 0.11952381  
## 0.010000000 0.2825714 0.10321429  
## 0.013035714 0.2687619 0.08595238  
## 0.024285714 0.2527619 0.06595238  
## 0.056428571 0.2161905 0.02023810  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.006071429.

#TFIDF  
#term frequency  
term.frequency <- function(row){  
 row / sum(row)  
 }  
  
#inverse document frequency  
inverse.doc.freq<- function(col){  
 corpus.size<- length(col)  
 doc.count<- length(which(col>0))  
 log10(corpus.size /doc.count)  
   
}  
  
tf.idf <- function(tf, idf){  
 tf\*idf  
}  
  
#normalize documents through TF  
train.tokens.df <- apply(train.tokens.matrix, 1, term.frequency)  
dim(train.tokens.df)

## [1] 6391 3500

view(train.tokens.df [1:20, 1:100])  
  
#Calculating the Inverse Document Frequency vector  
train.tokens.idf<-apply(train.tokens.matrix ,2, inverse.doc.freq)  
str(train.tokens.idf)

## Named num [1:6391] 0.852 1.451 1.386 1.077 2.845 ...  
## - attr(\*, "names")= chr [1:6391] "great" "batteri" "long" "buy" ...

#calculate tf-idf of our training data  
train.tokens.tfidf <- apply(train.tokens.df, 2, tf.idf, idf = train.tokens.idf)  
dim(train.tokens.tfidf)

## [1] 6391 3500

view(train.tokens.tfidf [1:25, 1:25])  
  
#transpose the matrix  
train.tokens.tfidf <- t(train.tokens.tfidf)  
dim(train.tokens.tfidf)

## [1] 3500 6391

view(train.tokens.tfidf [1:25, 1:25])  
  
#check for incomplete cases  
incomplete.cases<-which(!complete.cases(train.tokens.tfidf))  
train$review[incomplete.cases]

## [1] "It Is What It Is." "A+"

#Replace all in incomplete cases with a 0.0  
train.tokens.tfidf[incomplete.cases,]<- rep(0.0, ncol(train.tokens.tfidf))  
dim(train.tokens.tfidf)

## [1] 3500 6391

sum(which(!complete.cases(train.tokens.tfidf)))

## [1] 0

#Final tfidf data frame  
train.tokens.tfidf.df <- cbind(rating = train$rating, data.frame(train.tokens.tfidf))  
names(train.tokens.tfidf.df) <- make.names(names(train.tokens.tfidf.df))  
view(train.tokens.tfidf.df [1:25, 1:25])

set.seed(33445)  
cv.folds<-createMultiFolds(train$rating, k = 10, times = 3)  
  
cv.cntrl<- trainControl(method = "repeatedcv", number = 10, repeats = 3, index = cv.folds)  
  
#timing the code execution  
start.time <- Sys.time()  
  
# make a cluster to work on 8 logical cores  
cl<-makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
  
drops <- c("document")  
train.tokens.df <- train.tokens.df[ , !(names(train.tokens.df) %in% drops)]  
  
  
rpart.cv.2<- train(rating ~ ., data = train.tokens.tfidf.df, method = "rpart", trControl = cv.cntrl, tuneLength = 7)  
svmLinear3.cv.2<- train(rating ~., data = train.tokens.tfidf.df, method = "svmLinear3", trControl = cv.cntrl, tuneLength = 7)  
  
  
stopCluster(cl)  
  
#Execution time  
total.time2<- Sys.time() - start.time  
total.time2

## Time difference of 8.14405 mins

rpart.cv.1

## CART   
##   
## 3500 samples  
## 6391 predictors  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.006071429 0.3176190 0.14702381  
## 0.007678571 0.3007619 0.12595238  
## 0.007857143 0.2956190 0.11952381  
## 0.010000000 0.2825714 0.10321429  
## 0.013035714 0.2687619 0.08595238  
## 0.024285714 0.2527619 0.06595238  
## 0.056428571 0.2161905 0.02023810  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.006071429.

rpart.cv.2

## CART   
##   
## 3500 samples  
## 6391 predictors  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.006250000 0.3205714 0.15071429  
## 0.006785714 0.3174286 0.14678571  
## 0.007678571 0.3100952 0.13761905  
## 0.009821429 0.2919048 0.11488095  
## 0.017678571 0.2677143 0.08464286  
## 0.026071429 0.2505714 0.06321429  
## 0.052857143 0.2215238 0.02690476  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.00625.

svmLinear3.cv.1

## L2 Regularized Support Vector Machine (dual) with Linear Kernel   
##   
## 3500 samples  
## 6391 predictors  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## cost Loss Accuracy Kappa   
## 0.25 L1 0.3861905 0.2327381  
## 0.25 L2 0.3845714 0.2307143  
## 0.50 L1 0.3786667 0.2233333  
## 0.50 L2 0.3780952 0.2226190  
## 1.00 L1 0.3735238 0.2169048  
## 1.00 L2 0.3752381 0.2190476  
## 2.00 L1 0.3704762 0.2130952  
## 2.00 L2 0.3715238 0.2144048  
## 4.00 L1 0.3704762 0.2130952  
## 4.00 L2 0.3707619 0.2134524  
## 8.00 L1 0.3705714 0.2132143  
## 8.00 L2 0.3709524 0.2136905  
## 16.00 L1 0.3704762 0.2130952  
## 16.00 L2 0.3719048 0.2148810  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were cost = 0.25 and Loss = L1.

svmLinear3.cv.2

## L2 Regularized Support Vector Machine (dual) with Linear Kernel   
##   
## 3500 samples  
## 6391 predictors  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## cost Loss Accuracy Kappa   
## 0.25 L1 0.4033333 0.2541667  
## 0.25 L2 0.4075238 0.2594048  
## 0.50 L1 0.4012381 0.2515476  
## 0.50 L2 0.4029524 0.2536905  
## 1.00 L1 0.4009524 0.2511905  
## 1.00 L2 0.3954286 0.2442857  
## 2.00 L1 0.3946667 0.2433333  
## 2.00 L2 0.3858095 0.2322619  
## 4.00 L1 0.3875238 0.2344048  
## 4.00 L2 0.3718095 0.2147619  
## 8.00 L1 0.3753333 0.2191667  
## 8.00 L2 0.3603810 0.2004762  
## 16.00 L1 0.3639048 0.2048810  
## 16.00 L2 0.3548571 0.1935714  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were cost = 0.25 and Loss = L2.

#bi-gram (increasing the size of our matrix)  
train.tokens<- tokens\_ngrams(train.tokens, n = 1:2)  
  
train.tokens[[105]]

## [1] "part" "support" "obd1"   
## [4] "car" "flat" "lie"   
## [7] "test" "sever" "gm"   
## [10] "obd1" "car" "none"   
## [13] "work" "accur" "data"   
## [16] "error" "check" "can"   
## [19] "get" "fals" "differ"   
## [22] "data" "just" "select"   
## [25] "differ" "vehicl" "car"   
## [28] "differ" "set" "result"   
## [31] "firebird" "v8" "none"   
## [34] "accur" "compar" "real"   
## [37] "gm" "tech2" "scan"   
## [40] "tool" "select" "correct"   
## [43] "vehicl" "even" "work"   
## [46] "report" "invalid" "set"   
## [49] "detail" "investig" "reveal"   
## [52] "tool" "use" "gm"   
## [55] "truck" "spec" "firebird"   
## [58] "br" "br" "obd2"   
## [61] "perform" "correct" "featur"   
## [64] "limit" "can" "get"   
## [67] "featur" "elsewher" "less"   
## [70] "half" "cost" "scan"   
## [73] "tool" "real" "time"   
## [76] "sensor" "status" "data"   
## [79] "includ" "tool" "exampl"   
## [82] "can" "bought" "manufactur"   
## [85] "br" "br" "return"   
## [88] "prompt" "part\_support" "support\_obd1"   
## [91] "obd1\_car" "car\_flat" "flat\_lie"   
## [94] "lie\_test" "test\_sever" "sever\_gm"   
## [97] "gm\_obd1" "obd1\_car" "car\_none"   
## [100] "none\_work" "work\_accur" "accur\_data"   
## [103] "data\_error" "error\_check" "check\_can"   
## [106] "can\_get" "get\_fals" "fals\_differ"   
## [109] "differ\_data" "data\_just" "just\_select"   
## [112] "select\_differ" "differ\_vehicl" "vehicl\_car"   
## [115] "car\_differ" "differ\_set" "set\_result"   
## [118] "result\_firebird" "firebird\_v8" "v8\_none"   
## [121] "none\_accur" "accur\_compar" "compar\_real"   
## [124] "real\_gm" "gm\_tech2" "tech2\_scan"   
## [127] "scan\_tool" "tool\_select" "select\_correct"   
## [130] "correct\_vehicl" "vehicl\_even" "even\_work"   
## [133] "work\_report" "report\_invalid" "invalid\_set"   
## [136] "set\_detail" "detail\_investig" "investig\_reveal"   
## [139] "reveal\_tool" "tool\_use" "use\_gm"   
## [142] "gm\_truck" "truck\_spec" "spec\_firebird"   
## [145] "firebird\_br" "br\_br" "br\_obd2"   
## [148] "obd2\_perform" "perform\_correct" "correct\_featur"   
## [151] "featur\_limit" "limit\_can" "can\_get"   
## [154] "get\_featur" "featur\_elsewher" "elsewher\_less"   
## [157] "less\_half" "half\_cost" "cost\_scan"   
## [160] "scan\_tool" "tool\_real" "real\_time"   
## [163] "time\_sensor" "sensor\_status" "status\_data"   
## [166] "data\_includ" "includ\_tool" "tool\_exampl"   
## [169] "exampl\_can" "can\_bought" "bought\_manufactur"  
## [172] "manufactur\_br" "br\_br" "br\_return"   
## [175] "return\_prompt"

#transform to dfm and then a a matrix  
train.tokens.dfm <-dfm(train.tokens, tolower = FALSE)  
train.tokens.matrix <- as.matrix(train.tokens.dfm)  
train.tokens.dfm

## Document-feature matrix of: 3,500 documents, 68,370 features (99.9% sparse).  
## features  
## docs ya good qualiti guess trick stuck shape even rectangular fit  
## text1 0 1 0 0 0 0 0 0 0 0  
## text2 0 0 0 0 0 0 0 0 0 2  
## text3 0 0 0 0 0 0 0 0 0 0  
## text4 0 0 0 0 0 0 0 0 0 0  
## text5 0 0 0 0 0 0 0 0 0 0  
## text6 0 0 0 0 0 0 0 0 0 1  
## [ reached max\_ndoc ... 3,494 more documents, reached max\_nfeat ... 68,360 more features ]

#normalize all the documents via TF  
train.tokens.df<- apply(train.tokens.matrix, 1, term.frequency)  
  
  
  
#Calculating the Inverse Document Frequency vector  
train.tokens.idf<-apply(train.tokens.matrix ,2, inverse.doc.freq)  
  
#calculate tf-idf of our training data  
train.tokens.tfidf <- apply(train.tokens.df, 2, tf.idf, idf = train.tokens.idf)  
  
#transpose the matrix  
train.tokens.tfidf <- t(train.tokens.tfidf)  
  
#check for incomplete cases  
incomplete.cases<-which(!complete.cases(train.tokens.tfidf))  
train$review[incomplete.cases]

## [1] "It Is What It Is." "A+"

#Replace all in incomplete cases with a 0.0  
train.tokens.tfidf[incomplete.cases,]<- rep(0.0, ncol(train.tokens.tfidf))  
sum(which(!complete.cases(train.tokens.tfidf)))

## [1] 0

#Final tfidf data frame  
train.tokens.tfidf.df <- cbind(rating = train$rating, data.frame(train.tokens.tfidf))  
names(train.tokens.tfidf.df) <- make.names(names(train.tokens.tfidf.df))  
  
#applying LSA to extract relationships in our term-document Matrix and reduce dimensionality  
  
  
#Perform SVD to reduce dimentinality down to 300 columns  
#transpose our document term matrix to make it into a term document matrix to apply irlba function  
  
start.time<-Sys.time()  
train.irlba <-irlba(t(train.tokens.tfidf), nv = 200, maxit = 400)  
  
total.time3 <- Sys.time() -start.time  
total.time3

## Time difference of 11.84252 mins

train.svd<- data.frame(rating = train$rating, train.irlba$v)  
start.time<-Sys.time()  
  
cl<-makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
rf.cv.1<- train(rating ~ ., data = train.svd, method = "rf", trControl = cv.cntrl, tuneLength = 7)  
stopCluster(cl)  
total.time3 <- Sys.time() -start.time  
total.time3

## Time difference of 40.92469 mins

rf.cv.1

## Random Forest   
##   
## 3500 samples  
## 200 predictor  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.3485714 0.1857143  
## 35 0.3697143 0.2121429  
## 68 0.3720952 0.2151190  
## 101 0.3669524 0.2086905  
## 134 0.3687619 0.2109524  
## 167 0.3630476 0.2038095  
## 200 0.3687619 0.2109524  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 68.

#Adding text length feature to see if it improves our model  
train.svd$reviewlength <- train$reviewlength  
  
start.time<- Sys.time()  
  
cl<-makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
  
# Rerun the training with additional feature.  
  
rf.cv.2 <- train(rating ~ ., data = train.svd, method = "rf", trControl = cv.cntrl, tuneLength = 7, importance = TRUE)  
  
stopCluster(cl)  
  
totaltime5<- Sys.time() - start.time  
totaltime5

## Time difference of 1.299554 hours

confusionMatrix(train.svd$rating, rf.cv.2$finalModel$predicted)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 5 4 3 2 1  
## 5 308 167 84 90 51  
## 4 192 183 136 120 69  
## 3 66 147 193 174 120  
## 2 48 79 152 213 208  
## 1 28 53 77 188 354  
##   
## Overall Statistics  
##   
## Accuracy : 0.3574   
## 95% CI : (0.3415, 0.3736)  
## No Information Rate : 0.2291   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.1968   
##   
## Mcnemar's Test P-Value : 1.301e-06   
##   
## Statistics by Class:  
##   
## Class: 5 Class: 4 Class: 3 Class: 2 Class: 1  
## Sensitivity 0.4798 0.29094 0.30062 0.27134 0.4414  
## Specificity 0.8628 0.81992 0.82260 0.82063 0.8718  
## Pos Pred Value 0.4400 0.26143 0.27571 0.30429 0.5057  
## Neg Pred Value 0.8807 0.84071 0.83964 0.79571 0.8400  
## Prevalence 0.1834 0.17971 0.18343 0.22429 0.2291  
## Detection Rate 0.0880 0.05229 0.05514 0.06086 0.1011  
## Detection Prevalence 0.2000 0.20000 0.20000 0.20000 0.2000  
## Balanced Accuracy 0.6713 0.55543 0.56161 0.54598 0.6566

# using Linear SVM   
  
cl<-makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
  
start.time<- Sys.time()  
  
svmLinear3.cv.4<- train(rating ~., data = train.svd, method = "svmLinear3", trControl = cv.cntrl, tuneLength = 7)  
  
stopCluster(cl)  
  
totaltime4<- Sys.time() - start.time  
totaltime4

## Time difference of 23.06691 mins

svmLinear3.cv.4

## L2 Regularized Support Vector Machine (dual) with Linear Kernel   
##   
## 3500 samples  
## 201 predictor  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## cost Loss Accuracy Kappa   
## 0.25 L1 0.2303810 0.03797619  
## 0.25 L2 0.2312381 0.03904762  
## 0.50 L1 0.2380000 0.04750000  
## 0.50 L2 0.2312381 0.03904762  
## 1.00 L1 0.2365714 0.04571429  
## 1.00 L2 0.2312381 0.03904762  
## 2.00 L1 0.2444762 0.05559524  
## 2.00 L2 0.2312381 0.03904762  
## 4.00 L1 0.2433333 0.05416667  
## 4.00 L2 0.2312381 0.03904762  
## 8.00 L1 0.2352381 0.04404762  
## 8.00 L2 0.2312381 0.03904762  
## 16.00 L1 0.2462857 0.05785714  
## 16.00 L2 0.2312381 0.03904762  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were cost = 16 and Loss = L1.

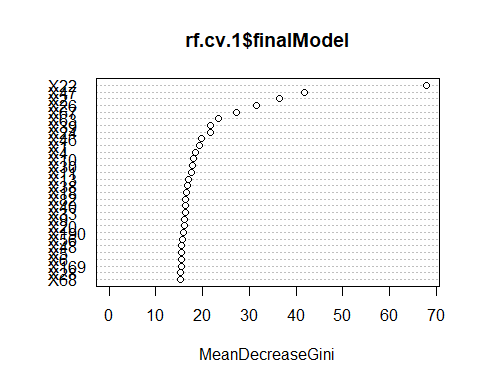
# using KNN   
  
cl <- makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
  
start.time<- Sys.time()  
  
knn.cv.2<- train(rating ~., data = train.svd, method = "knn", trControl = cv.cntrl, tuneLength = 7)  
  
stopCluster(cl)  
  
totaltime4<- Sys.time() - start.time  
totaltime4

## Time difference of 33.01372 secs

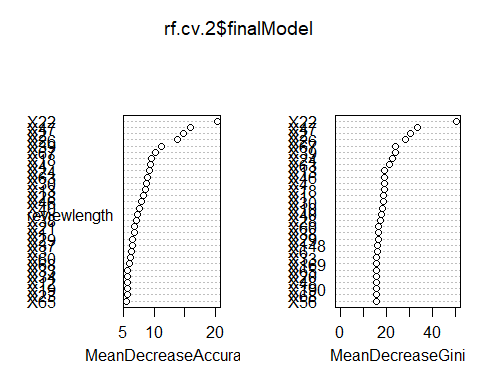
knn.cv.2

## k-Nearest Neighbors   
##   
## 3500 samples  
## 201 predictor  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 5 0.2207619 0.02595238  
## 7 0.2159048 0.01988095  
## 9 0.2186667 0.02333333  
## 11 0.2248571 0.03107143  
## 13 0.2155238 0.01940476  
## 15 0.2141905 0.01773810  
## 17 0.2146667 0.01833333  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 11.

#feature importance and feature engineering  
varImpPlot(rf.cv.1$finalModel)



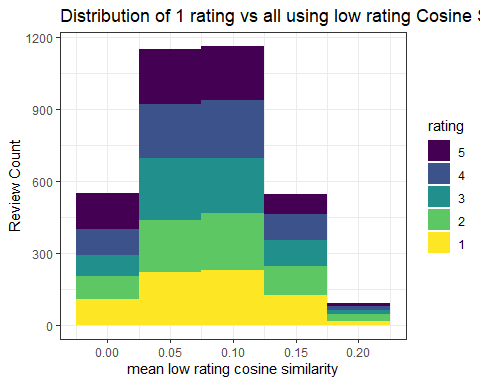
varImpPlot(rf.cv.2$finalModel)



#cosine similarity  
train.similarites <- cosine(t(as.matrix(train.svd[, -c(1,ncol(train.svd))])))  
  
dim(train.similarites)

## [1] 3500 3500

lowrating.indexes <- which(train$rating < "3")  
  
train.svd$lowratingsimilarities <- rep(0.0,nrow(train.svd))  
for(i in 1:nrow(train.svd)) {  
 train.svd$lowratingsimilarities[i] <- mean(train.similarites[i,lowrating.indexes])  
}  
  
ggplot(train.svd, aes(x =lowratingsimilarities, fill = rating))+ theme\_bw()+geom\_histogram(binwidth = 0.05) + labs(y= "Review Count", x= "mean low rating cosine similarity", title = "Distribution of 1 rating vs all using low rating Cosine Similarity")



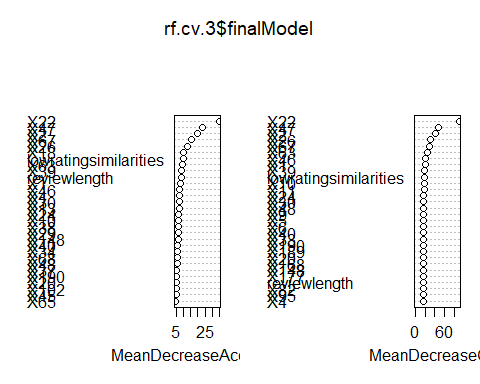
cl<-makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
start.time<- Sys.time()  
#Rerun the training with additional feature.  
rf.cv.3 <- train(rating ~ ., data = train.svd, method = "rf", trControl = cv.cntrl, tuneLength = 7, importance = TRUE)  
  
  
stopCluster(cl)  
  
totaltime6<- Sys.time() - start.time  
totaltime6

## Time difference of 1.203681 hours

confusionMatrix(train.svd$rating, rf.cv.3$finalModel$predicted)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 5 4 3 2 1  
## 5 319 163 78 72 68  
## 4 191 201 124 130 54  
## 3 67 122 213 176 122  
## 2 48 70 154 230 198  
## 1 40 52 74 190 344  
##   
## Overall Statistics  
##   
## Accuracy : 0.3734   
## 95% CI : (0.3574, 0.3897)  
## No Information Rate : 0.228   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.2168   
##   
## Mcnemar's Test P-Value : 1.14e-06   
##   
## Statistics by Class:  
##   
## Class: 5 Class: 4 Class: 3 Class: 2 Class: 1  
## Sensitivity 0.47970 0.33059 0.33126 0.28822 0.43766  
## Specificity 0.86561 0.82746 0.82954 0.82605 0.86883  
## Pos Pred Value 0.45571 0.28714 0.30429 0.32857 0.49143  
## Neg Pred Value 0.87643 0.85464 0.84643 0.79714 0.84214  
## Prevalence 0.19000 0.17371 0.18371 0.22800 0.22457  
## Detection Rate 0.09114 0.05743 0.06086 0.06571 0.09829  
## Detection Prevalence 0.20000 0.20000 0.20000 0.20000 0.20000  
## Balanced Accuracy 0.67265 0.57902 0.58040 0.55714 0.65324

varImpPlot(rf.cv.3$finalModel)



cl<-makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
start.time<- Sys.time()  
#Rerun the training with additional feature.  
rf.cv.2 <- train(rating ~ ., data = train.svd, method = "rf", trControl = cv.cntrl, tuneLength = 7, importance = TRUE)  
  
stopCluster(cl)  
  
totaltime5<- Sys.time() - start.time  
totaltime5

## Time difference of 1.176795 hours

confusionMatrix(train.svd$rating, rf.cv.2$finalModel$predicted)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 5 4 3 2 1  
## 5 335 150 71 88 56  
## 4 199 185 135 120 61  
## 3 75 127 191 181 126  
## 2 49 82 153 198 218  
## 1 32 55 70 195 348  
##   
## Overall Statistics  
##   
## Accuracy : 0.3591   
## 95% CI : (0.3432, 0.3753)  
## No Information Rate : 0.2311   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.1989   
##   
## Mcnemar's Test P-Value : 1.157e-07   
##   
## Statistics by Class:  
##   
## Class: 5 Class: 4 Class: 3 Class: 2 Class: 1  
## Sensitivity 0.48551 0.30885 0.30806 0.25320 0.43016  
## Specificity 0.87011 0.82248 0.82326 0.81531 0.86919  
## Pos Pred Value 0.47857 0.26429 0.27286 0.28286 0.49714  
## Neg Pred Value 0.87321 0.85214 0.84679 0.79143 0.83536  
## Prevalence 0.19714 0.17114 0.17714 0.22343 0.23114  
## Detection Rate 0.09571 0.05286 0.05457 0.05657 0.09943  
## Detection Prevalence 0.20000 0.20000 0.20000 0.20000 0.20000  
## Balanced Accuracy 0.67781 0.56566 0.56566 0.53425 0.64968

# using Linear SVM   
  
cl <- makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
  
start.time<- Sys.time()  
svmLinear3.cv.5<- train(rating ~., data = train.svd, method = "svmLinear3", trControl = cv.cntrl, tuneLength = 7)  
  
stopCluster(cl)  
  
totaltime4<- Sys.time() - start.time  
totaltime4

## Time difference of 23.66031 mins

svmLinear3.cv.5

## L2 Regularized Support Vector Machine (dual) with Linear Kernel   
##   
## 3500 samples  
## 202 predictor  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## cost Loss Accuracy Kappa   
## 0.25 L1 0.2413333 0.05166667  
## 0.25 L2 0.2313333 0.03916667  
## 0.50 L1 0.2334286 0.04178571  
## 0.50 L2 0.2313333 0.03916667  
## 1.00 L1 0.2375238 0.04690476  
## 1.00 L2 0.2313333 0.03916667  
## 2.00 L1 0.2407619 0.05095238  
## 2.00 L2 0.2313333 0.03916667  
## 4.00 L1 0.2394286 0.04928571  
## 4.00 L2 0.2313333 0.03916667  
## 8.00 L1 0.2418095 0.05226190  
## 8.00 L2 0.2313333 0.03916667  
## 16.00 L1 0.2412381 0.05154762  
## 16.00 L2 0.2313333 0.03916667  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were cost = 8 and Loss = L1.

# using KNN   
  
cl <- makeCluster(4, type = "SOCK")  
registerDoSNOW(cl)  
  
start.time<- Sys.time()  
  
knn.cv.3<- train(rating ~., data = train.svd, method = "knn", trControl = cv.cntrl, tuneLength = 7)  
  
stopCluster(cl)  
  
totaltime4<- Sys.time() - start.time  
totaltime4

## Time difference of 30.51882 secs

knn.cv.3

## k-Nearest Neighbors   
##   
## 3500 samples  
## 202 predictor  
## 5 classes: '5', '4', '3', '2', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3150, 3150, 3150, 3150, 3150, 3150, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 5 0.2182857 0.02285714  
## 7 0.2129524 0.01619048  
## 9 0.2182857 0.02285714  
## 11 0.2210476 0.02630952  
## 13 0.2170476 0.02130952  
## 15 0.2115238 0.01440476  
## 17 0.2146667 0.01833333  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 11.

test.tokens<- tokens(test$review, what = "word", remove\_numbers = TRUE, remove\_punct = TRUE, split\_hyphens = TRUE, remove\_symbols = TRUE)  
  
test.tokens <- tokens\_tolower(test.tokens)  
  
test.tokens <- tokens\_select(test.tokens, stopwords(), selection = "remove")  
  
test.tokens <- tokens\_wordstem(test.tokens, language = "english")  
  
test.tokens <- tokens\_ngrams(test.tokens, n = 1:2)  
  
test.tokens.dfm <- dfm(test.tokens, tolower = FALSE)  
  
  
train.tokens.dfm

## Document-feature matrix of: 3,500 documents, 68,370 features (99.9% sparse).  
## features  
## docs ya good qualiti guess trick stuck shape even rectangular fit  
## text1 0 1 0 0 0 0 0 0 0 0  
## text2 0 0 0 0 0 0 0 0 0 2  
## text3 0 0 0 0 0 0 0 0 0 0  
## text4 0 0 0 0 0 0 0 0 0 0  
## text5 0 0 0 0 0 0 0 0 0 0  
## text6 0 0 0 0 0 0 0 0 0 1  
## [ reached max\_ndoc ... 3,494 more documents, reached max\_nfeat ... 68,360 more features ]

test.tokens.dfm

## Document-feature matrix of: 1,500 documents, 31,187 features (99.9% sparse).  
## features  
## docs fit grill pair purchas yet anoth one fit\_grill grill\_pair pair\_purchas  
## text1 1 1 1 1 1 1 1 1 1 1  
## text2 0 0 0 0 0 0 0 0 0 0  
## text3 0 0 0 2 0 0 0 0 0 0  
## text4 0 0 0 0 0 0 0 0 0 0  
## text5 0 0 0 0 0 0 2 0 0 0  
## text6 0 0 1 1 0 0 0 0 0 0  
## [ reached max\_ndoc ... 1,494 more documents, reached max\_nfeat ... 31,177 more features ]

#Ensuring that TEST and TRAIN DFM have the same dimensions  
  
  
test.tokens.dfm <- dfm\_match(test.tokens.dfm, featnames(train.tokens.dfm))  
test.tokens.matrix<- as.matrix(test.tokens.dfm)  
test.tokens.dfm

## Document-feature matrix of: 1,500 documents, 68,370 features (100.0% sparse).  
## features  
## docs ya good qualiti guess trick stuck shape even rectangular fit  
## text1 0 0 0 0 0 0 0 0 0 1  
## text2 0 0 0 0 0 0 0 0 0 0  
## text3 0 0 0 0 0 0 0 0 0 0  
## text4 0 0 0 0 0 0 0 0 0 0  
## text5 0 0 0 0 0 0 0 0 0 0  
## text6 0 0 0 0 0 0 0 1 0 0  
## [ reached max\_ndoc ... 1,494 more documents, reached max\_nfeat ... 68,360 more features ]

#normalize the test dataset  
test.tokens.df <- apply(test.tokens.matrix, 1, term.frequency)  
str(test.tokens.df)

## num [1:68370, 1:1500] 0 0 0 0 0 ...  
## - attr(\*, "dimnames")=List of 2  
## ..$ features: chr [1:68370] "ya" "good" "qualiti" "guess" ...  
## ..$ docs : chr [1:1500] "text1" "text2" "text3" "text4" ...

#TFIDF conversion of the testdata  
  
  
  
test.tokens.tfidf <- apply(test.tokens.df, 2, tf.idf, idf = train.tokens.idf)  
  
test.tokens.tfidf <-t(test.tokens.tfidf)  
  
summary(test.tokens.tfidf[1,])

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000000 0.0000000 0.0000000 0.0000262 0.0000000 0.3937853

test.tokens.tfidf[is.na(test.tokens.tfidf)] <- 0.0  
summary(test.tokens.tfidf[1,])

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000000 0.0000000 0.0000000 0.0000262 0.0000000 0.3937853

#Applying SVD matrix factorization  
  
  
sigma.inverse <- 1 / train.irlba$d  
u.transpose <- t(train.irlba$u)  
  
test.svd.raw <- t(sigma.inverse \* u.transpose %\*% t(test.tokens.tfidf))  
dim(test.svd.raw)

## [1] 1500 200

test.svd <- data.frame(rating = test$rating, test.svd.raw, reviewlength = test$reviewlength)  
  
  
  
test.similarities<- rbind(test.svd.raw, train.irlba$v[lowrating.indexes,])  
test.similarities<- cosine(t(test.similarities))  
  
  
  
  
  
test.svd$lowratingsimilarities <- rep(0.0, nrow(test.svd))  
lowrating.cols <- (nrow(test.svd) + 1):ncol(test.similarities)  
for(i in 1:nrow(test.svd)) {  
 test.svd$lowratingsimilarities[i] <- mean(test.similarities[i, lowrating.cols])   
}  
  
  
  
  
test.svd$lowratingsimilarities[!is.finite(test.svd$lowratingsimilarities)] <- 0  
  
  
  
  
preds<- predict(rf.cv.3, test.svd)  
  
confusionMatrix(preds, test.svd$rating)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 5 4 3 2 1  
## 5 146 80 45 34 19  
## 4 65 89 38 21 17  
## 3 37 61 85 55 41  
## 2 28 45 93 92 60  
## 1 24 25 39 98 163  
##   
## Overall Statistics  
##   
## Accuracy : 0.3833   
## 95% CI : (0.3586, 0.4085)  
## No Information Rate : 0.2   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.2292   
##   
## Mcnemar's Test P-Value : 3.743e-05   
##   
## Statistics by Class:  
##   
## Class: 5 Class: 4 Class: 3 Class: 2 Class: 1  
## Sensitivity 0.48667 0.29667 0.28333 0.30667 0.5433  
## Specificity 0.85167 0.88250 0.83833 0.81167 0.8450  
## Pos Pred Value 0.45062 0.38696 0.30466 0.28931 0.4670  
## Neg Pred Value 0.86905 0.83386 0.82391 0.82403 0.8810  
## Prevalence 0.20000 0.20000 0.20000 0.20000 0.2000  
## Detection Rate 0.09733 0.05933 0.05667 0.06133 0.1087  
## Detection Prevalence 0.21600 0.15333 0.18600 0.21200 0.2327  
## Balanced Accuracy 0.66917 0.58958 0.56083 0.55917 0.6942

preds1<- predict(svmLinear3.cv.5, test.svd)  
confusionMatrix(preds1,test.svd$rating)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 5 4 3 2 1  
## 5 246 219 205 198 184  
## 4 17 18 9 2 2  
## 3 4 8 11 7 2  
## 2 1 0 1 7 4  
## 1 32 55 74 86 108  
##   
## Overall Statistics  
##   
## Accuracy : 0.26   
## 95% CI : (0.238, 0.283)  
## No Information Rate : 0.2   
## P-Value [Acc > NIR] : 1.107e-08   
##   
## Kappa : 0.075   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: 5 Class: 4 Class: 3 Class: 2 Class: 1  
## Sensitivity 0.8200 0.0600 0.036667 0.023333 0.3600  
## Specificity 0.3283 0.9750 0.982500 0.995000 0.7942  
## Pos Pred Value 0.2338 0.3750 0.343750 0.538462 0.3042  
## Neg Pred Value 0.8795 0.8058 0.803134 0.802959 0.8323  
## Prevalence 0.2000 0.2000 0.200000 0.200000 0.2000  
## Detection Rate 0.1640 0.0120 0.007333 0.004667 0.0720  
## Detection Prevalence 0.7013 0.0320 0.021333 0.008667 0.2367  
## Balanced Accuracy 0.5742 0.5175 0.509583 0.509167 0.5771

pred2<- predict(knn.cv.3, test.svd)  
confusionMatrix(pred2, test.svd$rating)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 5 4 3 2 1  
## 5 67 65 59 57 62  
## 4 67 73 60 61 61  
## 3 51 54 62 57 63  
## 2 59 52 62 54 56  
## 1 56 56 57 71 58  
##   
## Overall Statistics  
##   
## Accuracy : 0.2093   
## 95% CI : (0.189, 0.2308)  
## No Information Rate : 0.2   
## P-Value [Acc > NIR] : 0.1913   
##   
## Kappa : 0.0117   
##   
## Mcnemar's Test P-Value : 0.9231   
##   
## Statistics by Class:  
##   
## Class: 5 Class: 4 Class: 3 Class: 2 Class: 1  
## Sensitivity 0.22333 0.24333 0.20667 0.1800 0.19333  
## Specificity 0.79750 0.79250 0.81250 0.8092 0.80000  
## Pos Pred Value 0.21613 0.22671 0.21603 0.1908 0.19463  
## Neg Pred Value 0.80420 0.80730 0.80379 0.7979 0.79867  
## Prevalence 0.20000 0.20000 0.20000 0.2000 0.20000  
## Detection Rate 0.04467 0.04867 0.04133 0.0360 0.03867  
## Detection Prevalence 0.20667 0.21467 0.19133 0.1887 0.19867  
## Balanced Accuracy 0.51042 0.51792 0.50958 0.4946 0.49667