Text mining project

mindu931

16 January 2019

# Abstract

Wine sensory audition and assessment is never easy, even with a wine specialist. However, the description which is printed on each bottle can bring us some helpful information. Base on that ideal, I decided to analyze the data about wine review (take form Kaggle) and help the consumers make the decision on two different aspects. The first one is which is a good and excellent wine. The second aspect is which is the worth the price bottle (high value with a limit amount of money). I use different methods to process the text data, then when compare Naive Bayes, SVM linear and SVM RBF kernel when building the best model for predicting. The final result is quite impressive with accuracy aa%. Base on this model, some suggestions are provided for the consumer to choose a satisfied bottle.

# Introduction

## Motivtion

I always confuse when standing in a wine cellar. How can I choose an excellent wine for my family? What is the best wine I can choose with my budget? The problem can be solved easier if I had a good sense. But, sadly, I don’t, like many people in this world. The problems still continue when I listen to the advice of the salesman and even taste some wine. So, I usually choose randomly one bottle based on these devices for my event. Unluckily, sometimes my relatives don’t think that it is a good wine. ## Aim I guess that the information from the provider (description, country, region, designation…) should bring some important information. So, I try to solve the problems base on text mining ideal: Take the wine data which is already reviewed and marked by some specialists, then process these data together with the point and build a model to prediction in order to figure out which bottle is good or excellent, which one have high benefit or just medium. ## Content of this report In this report, Firstly, I will provide information about the data as well as the way I define the class for each bottle. Then, I briefly summary relevant theory. The method I used will be presented in details. It includes the steps for preprocessing, compares and choosing the best model, my suggestions for choosing wine. Finally is some discussions and my conclusion.

# Data

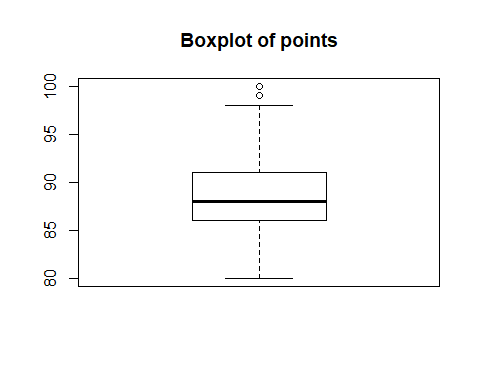
Wine review data is taken form Kaagle [link](https://www.kaggle.com/zynicide/wine-reviews). I choose the second .csv version of this dataset. In this version, duplicated data is removed. The data include 120975 samples about wine. Each sample contains the following information: country, description, designation, points, price, province, region, more specific region (region2), tester name, taster\_twitter handle, title, variety, and winery. Here is the overlook of the review:

wine <- read.csv("C:/Users/Duong Minh Duc/Documents/GitHub/Text-Mining-Project/wine.csv")  
head(wine)

## X country  
## 1 0 Italy  
## 2 1 Portugal  
## 3 2 US  
## 4 3 US  
## 5 4 US  
## 6 5 Spain  
## description  
## 1 Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity.  
## 2 This is ripe and fruity, a wine that is smooth while still structured. Firm tannins are filled out with juicy red berry fruits and freshened with acidity. It's already drinkable, although it will certainly be better from 2016.  
## 3 Tart and snappy, the flavors of lime flesh and rind dominate. Some green pineapple pokes through, with crisp acidity underscoring the flavors. The wine was all stainless-steel fermented.  
## 4 Pineapple rind, lemon pith and orange blossom start off the aromas. The palate is a bit more opulent, with notes of honey-drizzled guava and mango giving way to a slightly astringent, semidry finish.  
## 5 Much like the regular bottling from 2012, this comes across as rather rough and tannic, with rustic, earthy, herbal characteristics. Nonetheless, if you think of it as a pleasantly unfussy country wine, it's a good companion to a hearty winter stew.  
## 6 Blackberry and raspberry aromas show a typical Navarran whiff of green herbs and, in this case, horseradish. In the mouth, this is fairly full bodied, with tomatoey acidity. Spicy, herbal flavors complement dark plum fruit, while the finish is fresh but grabby.  
## designation points price province  
## 1 VulkÃ  Bianco 87 NA Sicily & Sardinia  
## 2 Avidagos 87 15 Douro  
## 3 87 14 Oregon  
## 4 Reserve Late Harvest 87 13 Michigan  
## 5 Vintner's Reserve Wild Child Block 87 65 Oregon  
## 6 Ars In Vitro 87 15 Northern Spain  
## region\_1 region\_2 taster\_name  
## 1 Etna Kerin Oâ\200\231Keefe  
## 2 Roger Voss  
## 3 Willamette Valley Willamette Valley Paul Gregutt  
## 4 Lake Michigan Shore Alexander Peartree  
## 5 Willamette Valley Willamette Valley Paul Gregutt  
## 6 Navarra Michael Schachner  
## taster\_twitter\_handle  
## 1 @kerinokeefe  
## 2 @vossroger  
## 3 @paulgwineÂ   
## 4   
## 5 @paulgwineÂ   
## 6 @wineschach  
## title  
## 1 Nicosia 2013 VulkÃ  Bianco (Etna)  
## 2 Quinta dos Avidagos 2011 Avidagos Red (Douro)  
## 3 Rainstorm 2013 Pinot Gris (Willamette Valley)  
## 4 St. Julian 2013 Reserve Late Harvest Riesling (Lake Michigan Shore)  
## 5 Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)  
## 6 Tandem 2011 Ars In Vitro Tempranillo-Merlot (Navarra)  
## variety winery  
## 1 White Blend Nicosia  
## 2 Portuguese Red Quinta dos Avidagos  
## 3 Pinot Gris Rainstorm  
## 4 Riesling St. Julian  
## 5 Pinot Noir Sweet Cheeks  
## 6 Tempranillo-Merlot Tandem

Points are important information about this data. Points are marked by wine specialist, which make it reliable. The owner only public data which have at least 80 point (out of 100) which means the data contain only good and excellent wine. Here is the distribution of the point

boxplot(wine$points, main="Boxplot of points")



#bar plot

and some plots about other informations:…

## Theory

In this section, I will briefly talk about some theory in nature language processing(NLP) and machine learning(ML) in a simple way of reviewing. For some terms, only the aspect that relevant to this project is presented. I assume that the reader already has some basic knowledge about NLP and ML before(this part is not a guide for a person with a blank background). I also noticed that we are processing the text written in English.

-Terms in preprocessing: + Stop words: This is the term for useless words in the text. To be specific, these words usually have no (or very low) meaning but represent a lot. For example: a, an, the… When processing text data. We usually remove these words. + Steaming: This is the process of reducing a word to its root form. For example: processes, processing and processed are 3 different words, but it just 3 different representatives of the word “process”. Another example is evaluated and evaluation. + tokenization: the process of splitting text into smaller parts. Each part can consider as a feature when training in machine learning kernels. If the smaller part here is the single word (split the text to single words) then all words we have will become the bag of words. + Corpus: You can understand simply that corpus is a collection of all features in NLP, used for training. For example, all word in the bag of words is a corpus. + Document term matrix(Dtm) is a matrix, which has column is all words of the corpus and column is the document ID. The cross between row and column is the point for that word in that document. Points can be calculated in some different ways. In this project, I use tf-idf. + tf-idf: Stand for term frequency-inverse document frequency. This is a statistic method for calculating the importance of each word following this formula:

number of documents where the term t appers. If this value equal 0, it will be adjust to 1 D: total number of documents in the corpus

To be specific, If a word is represented many times in almost documents, it is not so importance, and vice versa.

* Machine learning kernels: a kernel can understand as a core algorithm used when training a model in ML. Here are the three kernels that I use in this report:

+Naive Bayes: Let initial with we have a vector X of a document, which builds in document-term matrix. The probability of assign a class label y to that vector is:

Naive-Bayes theory make an asump that all features are independent. The assumption is not realistic, somehow, it works well.

+SVM: stand for support vector machine. In the previous kernel, we have vector X for each document. ++SVM linear…

++SVM RBF…

-Ngram:

-Pos of tag:

## Method

In this part, I’ll talk about the method for the first problem of this project: Which is the “good” and “excellent” wine. The second problem about wine that has “high” and “medium” value is solved using a similar method but will be discussed in details in a different part. - Define the class: The first step is to define the class for each document. Base on the distribution of the point, I will take the point 88 as the boundary because it is the median value, which will make the data become balance. So, the wine that has higher than 88 will be the excellent wine, while the rest is the good wine. - Preprocess data + Split train and test: I use 80% of the data as the training set and 20% as the testing set. Then, the training data is processed as follow: + Make the text ready: I decided that the information about the country, designation, province, region, and wine variety is important. So, I decided to merge that information into the description as a paragraph. That paragraph is the one that I will process. + Language convert: In the data set. there is some name of variety that is not unified (same type but written in the different language). So I convert it to the most famous word as follow: ++ Replace German names with English names: “weissburgunder” is replaced as “chardonnay”. “spatburgunder” is replaced as “pinot noir”. “grauburgunder” is replaced as “pinot gris”.  
++ Replace the Spanish “garnacha”" with the french “grenache”. ++ Replace the Italian “pinot nero”" with the french “pinot noir”. ++ Replace the Portugues “alvarinho”" with the spanish “albarino”.

* Remove non-ASCII characters.
* Remove punction .
* Make words to lower from.
* Remove number.
* Remove stop words: I use the default stop words lists in the tm library and then adjust it. Firstly, the lists have the word “very”. In my opinion, very is a valuable word in this data. Because in the descriptions, for example, “sweet” and “very sweet” are different levels of flavor. So, I remove that word on the list. Then, I add three words “the”, “and” and “wine” to the stop word list because, in this data set, it doesn’t have any meaning. Here is the final stop word list:

stopwords <- stopwords("english")  
stopwords <- stopwords[!stopwords=="very"]  
stopwords <- c("the", "and", "wine", stopwords)  
  
stopwords

## [1] "the" "and" "wine" "i" "me"   
## [6] "my" "myself" "we" "our" "ours"   
## [11] "ourselves" "you" "your" "yours" "yourself"   
## [16] "yourselves" "he" "him" "his" "himself"   
## [21] "she" "her" "hers" "herself" "it"   
## [26] "its" "itself" "they" "them" "their"   
## [31] "theirs" "themselves" "what" "which" "who"   
## [36] "whom" "this" "that" "these" "those"   
## [41] "am" "is" "are" "was" "were"   
## [46] "be" "been" "being" "have" "has"   
## [51] "had" "having" "do" "does" "did"   
## [56] "doing" "would" "should" "could" "ought"   
## [61] "i'm" "you're" "he's" "she's" "it's"   
## [66] "we're" "they're" "i've" "you've" "we've"   
## [71] "they've" "i'd" "you'd" "he'd" "she'd"   
## [76] "we'd" "they'd" "i'll" "you'll" "he'll"   
## [81] "she'll" "we'll" "they'll" "isn't" "aren't"   
## [86] "wasn't" "weren't" "hasn't" "haven't" "hadn't"   
## [91] "doesn't" "don't" "didn't" "won't" "wouldn't"   
## [96] "shan't" "shouldn't" "can't" "cannot" "couldn't"   
## [101] "mustn't" "let's" "that's" "who's" "what's"   
## [106] "here's" "there's" "when's" "where's" "why's"   
## [111] "how's" "a" "an" "the" "and"   
## [116] "but" "if" "or" "because" "as"   
## [121] "until" "while" "of" "at" "by"   
## [126] "for" "with" "about" "against" "between"   
## [131] "into" "through" "during" "before" "after"   
## [136] "above" "below" "to" "from" "up"   
## [141] "down" "in" "out" "on" "off"   
## [146] "over" "under" "again" "further" "then"   
## [151] "once" "here" "there" "when" "where"   
## [156] "why" "how" "all" "any" "both"   
## [161] "each" "few" "more" "most" "other"   
## [166] "some" "such" "no" "nor" "not"   
## [171] "only" "own" "same" "so" "than"   
## [176] "too"

+Steamming: I steam the word by using SnowballC steam.  
+Tonkenzine and making the bag of words as the corpus. Then I only keep 99% spare term. Which mean that only the words that appear in at least 1% of documents are kept. It makes sense because there are some rare words which only appear in one or a few documents. That words are not helpful for training because maybe we never meet it again in the testing set. +Create the document terms matrix base on tf-idf. +Now the data is ready for training. For testing data, the preprocess is similar, except the Dtm. Dtm of testing data will be create based on the corpus of training data (which means some words that don’t appear in the training data will be drop)

-Compare kernels: At this step, I’ll use the ready Dtm for training model, then test with the testing Dtm. Three kernel Naive-Bayes, SVM linear and SVM RBF are used to compare. In this step, because of the limit in my resource (Laptop core i5 7th gen, 8GB) and time. I only use the sample 20% of the data (16% - 19356 samples for training and 4% - 4839 samples for testing) to compute. Different parameters used for training is also reported. Notice that the time for training NB and SVM linear is quite fast (a few minutes) but It takes a long time for training SVM RBF (~14 hours, for 505 terms in Dtm include time for tunning) - Improve the best model: After comparing, the best model is selected. Then I continue to improve the model when using n-gram and adjust the weight for some terms using Part-Of-Speech Tagger (POS Tagger). - Final result: Final model is selected and run again with 100% data. The final result is reported based on this model.

## Result and explain

* Firstly, Let’s takes a look at 5 original paragraphs and the paragraphs after preprocessing to see how it work:

# Use for check the not available and duplicate data, but not necessary  
#wine <- na.omit(wine)  
#wine[duplicated(wine),]  
wine$quality <- wine$points > 88  
wine$quality[wine$quality == TRUE] <- "excellent"  
wine$quality[wine$quality == FALSE] <- "good"  
wine$quality <- as.factor(wine$quality)  
wine$value <- wine$points/log(wine$price)  
wine$benefit <- wine$value > 27  
wine$benefit[wine$benefit == TRUE] <- "high"  
wine$benefit[wine$benefit == FALSE] <- "medium"  
wine$benefit <- as.factor(wine$benefit)  
wine$description <- paste(wine$description, wine$country, wine$designation, wine$province, wine$region\_1, wine$region\_2, wine$variety)  
wine$description <- as.character(wine$description)  
  
wine$description <- gsub("weissburgunder", "chardonnay", wine$description)  
wine$description <- gsub("spatburgunder", "pinot noir", wine$description)  
wine$description <- gsub("grauburgunder", "pinot gris", wine$description)  
  
#Replace the Spanish garnacha with the french grenache  
wine$description <- gsub("garnacha", "grenache", wine$description)  
  
#Replace the Italian pinot nero with the french pinot noir  
wine$description <- gsub("pinot nero", "pinot noir", wine$description)  
  
#Replace the Portugues alvarinho with the spanish albarino  
wine$description <- gsub("alvarinho", "albarino", wine$description)  
  
#clean function  
wine$description <- iconv(wine$description, from = "UTF-8", to = "ASCII", sub = "")

n = dim(wine)[1]  
set.seed(12345)  
id = sample(1:n, floor(n\*0.8))  
train\_full = wine[id,]  
test\_full = wine[-id,]  
  
id2 = sample(1:n, floor(n\*0.2))  
wine\_sample <- wine[id2,]  
wine2 <- wine[id2,]  
n2 = length(id2)  
id\_test = sample(1:n2, floor(n2\*0.8))  
train = wine\_sample[id\_test,]  
test = wine\_sample[-id\_test,]  
  
##clean function  
clean <- function(text\_vector)  
 {  
 wine\_corpus = VCorpus(VectorSource(text\_vector))  
 wine\_corpus = tm\_map(wine\_corpus, removePunctuation)  
 wine\_corpus = tm\_map(wine\_corpus, content\_transformer(tolower))  
 wine\_corpus = tm\_map(wine\_corpus, removeNumbers)  
 wine\_corpus = tm\_map(wine\_corpus, removeWords, stopwords )  
 #wine\_corpus = tm\_map(wine\_corpus, stripWhitespace)  
 wine\_corpus <- tm\_map(wine\_corpus, stemDocument)  
   
   
 return(wine\_corpus)  
 }  
  
##create the train set  
wine\_train\_set <- clean(train$description)  
  
train$description[1:5]

## [1] "A crush of raspberry and maraschino-cherry perfume persists throughout this pristine pretty Pinot Noir. Punchy red-fruit flavors are concentrated and fresh, gilded by subtle tones of violet, toast and crushed mineral. The finish is long, framed neatly by penetrating but finely grained tannins. US New York Cayuga Lake Finger Lakes Pinot Noir"   
## [2] "This wine is a blend of all six Bordeaux varieties: 35% Merlot, 30% Cabernet Sauvignon, 25% Cabernet Franc, 4% Carmnre, 3% Malbec and 3% Petit Verdot. Its coffee, cranberry, herb, currant, barrel spice and anise aromas lead to tightly wound, layered red and black fruit flavors. Plump, silky tannins provide the frame. US Washington State Cuve Washington Washington Washington Other Bordeaux-style Red Blend"  
## [3] "Black as night in the glass and noticeably viscous, this shows brooding aromas of heavy char, black tobacco, cassis, bloody meat and balsamic vinegar. The palate isn't so heavy, though. Good acidity and tannins glide across the tongue while offering flavors of pencil lead, black cherry extract and black pepper. US Blessings California Santa Clara Valley Central Coast Petite Sirah"   
## [4] "Black cherry fruit gets serious with elements of slate, eucalyptus, tar and wild mint on this wine from the Corralitos area at the southern coastal edge of the appellation. It's soft at first on the palate, then reveals a wash of acidity, with sour cherry fruit and black tea tannins. US Alfaro Family Vineyards California Santa Cruz Mountains Central Coast Pinot Noir"   
## [5] "Opaque and inky with shades of purple and black, this dessert wine smells both fresh and dense with blackberry, blueberry and gravel notes. The palate packs ripe blueberry and black-cherry-paste flavors with grippy plum-skin texture and a touch of espresso, proving quite sweet but complex. US Tranquilo California Paso Robles Central Coast Petite Sirah"

wine\_train\_set[1:5]

## <<VCorpus>>  
## Metadata: corpus specific: 0, document level (indexed): 0  
## Content: documents: 5

* Then, Here is the Dtm for the first sentence of training set:

train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set, control = list(weighting = weightTfIdf))  
#train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set, control = list( tokenize=NLP\_tokenizer))  
#train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set)  
train\_dtm\_tfidf <- removeSparseTerms(train\_dtm\_tfidf, 0.99)  
  
#wine\_train\_set <- cbind(wine\_train\_set, train$quality)  
  
#create the test set  
wine\_test\_set <- clean(test$description)  
wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) ,weighting = weightTfIdf))  
#wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) , tokenize=NLP\_tokenizer))  
  
#create matrix for training  
wine\_train\_set <<- as.matrix(train\_dtm\_tfidf)  
wine\_test\_set <- as.matrix(wine\_test\_set)  
wine\_test\_set <- wine\_test\_set[,Terms(train\_dtm\_tfidf)]  
#create the test result  
wine\_testing\_result <- test$quality  
  
  
train\_dtm\_tfidf[1,]

And let see the first line of Dtm for testing set see make sure that the prepared data is correct. As we can see, the term remains. Just the points are different.

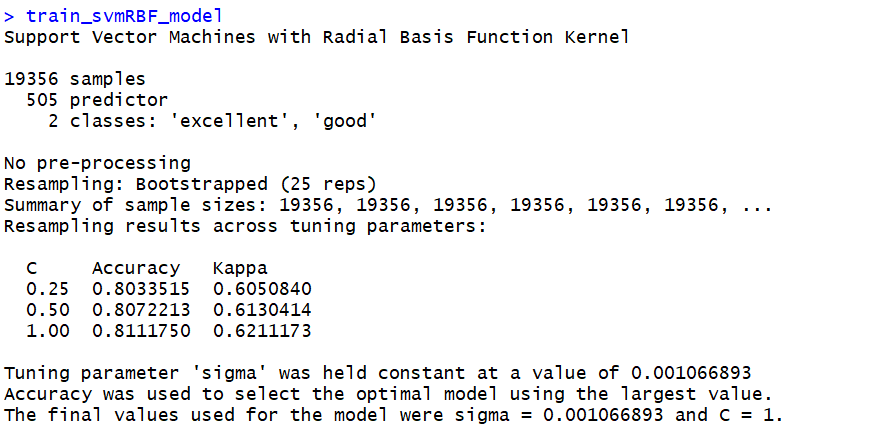
wine\_test\_set[1,]

Now, the data is ready. I train that dataset with threes different kernels as mentioned. And here is the results:

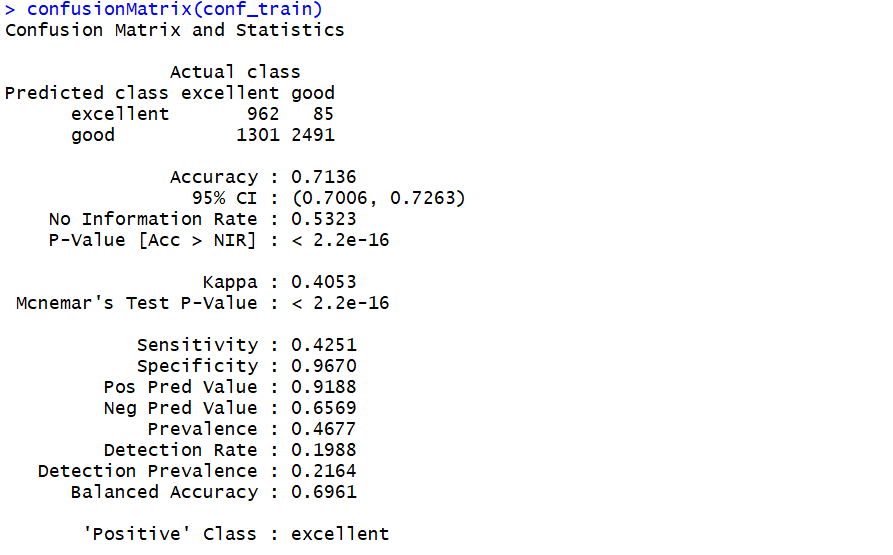
train\_nb\_model <- train(x= wine\_train\_set, y=train$quality , method = 'naive\_bayes')  
train\_nb\_model  
model\_nb\_result <- predict(train\_nb\_model, newdata = wine\_test\_set)  
  
conf\_nb\_train <- table(model\_nb\_result, wine\_testing\_result)  
names(dimnames(conf\_nb\_train)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_nb\_train)

#Here is the SVM Linear kenel  
train\_svmLinear\_model <- train(x= wine\_train\_set, y=train$quality , method = 'svmLinear3')  
train\_svmLinear\_model  
model\_svmLinear\_result <- predict(train\_svmLinear\_model, newdata = wine\_test\_set)  
  
conf\_svmLinear\_train <- table(model\_svmLinear\_result, wine\_testing\_result)  
names(dimnames(conf\_svmLinear\_train)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_svmLinear\_train)

# #Here is the SVM RBF kenel  
# # Because the trainning time is long. I'll not run this code and only attach images of the previous run.  
# train\_svmRBF\_model <- train(x= wine\_train\_set, y=train$quality , method = 'svmRadial')  
# train\_svmRBF\_model  
# model\_svmRBF\_result <- predict(train\_svmRBF\_model, newdata = wine\_test\_set)  
#   
# conf\_svmRBF\_train <- table(model\_svmRBF\_result, wine\_testing\_result)  
# names(dimnames(conf\_svmRBF\_train)) <- c("Predicted class", "Actual class")  
# confusionMatrix(conf\_svmRBF\_train)



SVMRBF kenel



SVMRBF result

As we can see in the result, The accuracy when training data for NB, SVM Linear and SVM RBF is aa%, bb%, and 81.11%. As a result, I expected that the accuracy when testing with test data will be similar. But it just 74.68% for NB, 78.59% for SVM Linear and 71.36%. SVM Linear is the kernel which has the highest value of accuracy. No information rate (NIR) 0.5323 mean that a class takes 53.23%(Good class), which mean the data is balanced. We can judge that the model is actually worked. But, accuracy is just one side of the story. Let see about the classification: We have two classes “good” and “excellent”. If the class is predicted exactly, it’s perfect. Obviously, excellent wine is better than good wine. So, if a good wine is predicted as excellent wine, it’s hard to accept (Similar to False negative). In contrast, If you pretend to buy good wine but have excellent wine. You are just lucky and nothing happened. (similar with False positive) SVM RBF kernel show the lowest number of False negative. But, I also see that the predicted result is biased to the “good class”: The number of good wine is predicted is triple as the number of excellent wine. It’s quite hard for understand. On the other hand, SVM Linear is better than NB in all indicators.

After all, I consider the accuracy, the number of confusion matrix and the training time for choosing the best model. In my opinion, It’s SVM Linear. (SVM RBF is interesting but it’s hard when I try to improve the model with that high training time, consider the scope of this project)

-Improve the model: I’ll try to improve the SVM linear model with the folling methods: + Improve with n-gram: Firtsly, I try to train the model with 2-gram, (the preprocess still remain) here is the first line of Dtm:

NLP\_tokenizer <- function(x) {  
 unlist(lapply(ngrams(words(x), 2:2), paste, collapse = "\_"), use.names = FALSE)  
}  
  
wine\_train\_set <- clean(train$description)  
  
train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set, control = list(weighting = weightTfIdf, tokenize=NLP\_tokenizer))  
#train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set, control = list( tokenize=NLP\_tokenizer))  
#train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set)  
train\_dtm\_tfidf <- removeSparseTerms(train\_dtm\_tfidf, 0.99)  
  
  
NLP\_tokenizer <- function(x) {  
 unlist(lapply(ngrams(words(x), 2:2), paste, collapse = "\_"), use.names = FALSE)  
}  
  
#create the test set  
wine\_test\_set <- clean(test$description)  
wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) ,weighting = weightTfIdf, tokenize=NLP\_tokenizer))  
#wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) , tokenize=NLP\_tokenizer))  
  
#create matrix for training  
wine\_train\_set <<- as.matrix(train\_dtm\_tfidf)  
wine\_test\_set <- as.matrix(wine\_test\_set)  
wine\_test\_set <- wine\_test\_set[,Terms(train\_dtm\_tfidf)]  
#create the test result  
wine\_testing\_result <- test$quality  
  
wine\_train\_set[1,]

result:

train\_svmLinear\_model <- train(x= wine\_train\_set, y=train$quality , method = 'svmLinear3')  
train\_svmLinear\_model  
model\_svmLinear\_result <- predict(train\_svmLinear\_model, newdata = wine\_test\_set)  
  
conf\_svmLinear\_train <- table(model\_svmLinear\_result, wine\_testing\_result)  
names(dimnames(conf\_svmLinear\_train)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_svmLinear\_train)  
  
train\_svm\_model <- train(x= wine\_train\_set, y=train$quality , method = 'svmLinear3')

The result is quite bad, which means 2-gram is not usuful. Based on my observation. After preprocess and steaming, the description is quite discrate. I means the words is likely not connected to each other. It lead to 2-gram model not works.

* I continue with both bag of words and 2-gram (which means 1-2gram). Here is the first line of Dtm:

NLP\_tokenizer <- function(x) {  
 unlist(lapply(ngrams(words(x), 1:2), paste, collapse = "\_"), use.names = FALSE)  
}  
  
wine\_train\_set <- clean(train$description)  
  
train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set, control = list(weighting = weightTfIdf, tokenize=NLP\_tokenizer))  
#train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set, control = list( tokenize=NLP\_tokenizer))  
#train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set)  
train\_dtm\_tfidf <- removeSparseTerms(train\_dtm\_tfidf, 0.99)  
  
  
NLP\_tokenizer <- function(x) {  
 unlist(lapply(ngrams(words(x), 1:1), paste, collapse = "\_"), use.names = FALSE)  
}  
  
#create the test set  
wine\_test\_set <- clean(test$description)  
wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) ,weighting = weightTfIdf, tokenize=NLP\_tokenizer))  
#wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) , tokenize=NLP\_tokenizer))  
  
#create matrix for training  
wine\_train\_set <<- as.matrix(train\_dtm\_tfidf)  
wine\_test\_set <- as.matrix(wine\_test\_set)  
wine\_test\_set <- wine\_test\_set[,Terms(train\_dtm\_tfidf)]  
#create the test result  
wine\_testing\_result <- test$quality  
  
wine\_train\_set[1,]

And here is the result:

train\_svmLinear\_model <- train(x= wine\_train\_set, y=train$quality , method = 'svmLinear3')  
train\_svmLinear\_model  
model\_svmLinear\_result <- predict(train\_svmLinear\_model, newdata = wine\_test\_set)  
  
conf\_svmLinear\_train <- table(model\_svmLinear\_result, wine\_testing\_result)  
names(dimnames(conf\_svmLinear\_train)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_svmLinear\_train)

The accurency is is slightly decreses. Good new is the False negative is slightly better. It’s like the trade off between models. Consider that and the dimension of Dtm is large (which lead to high trainning time), while there are too much features with 0 point. I conclude that 1-2 gram is not so useful. Overall, n-gram is not a way to improve the model.

* An other attempt to improve is adjust the weight for Dtm. As I obserse in the data. Description usually talk about ingerdiens: lemon, cherry or region: Veneto… So, the first idea is double the point for nouns and see how it works. On the other hand, Adjactive used for drescribe the flavor of wine could make an important role. So, the second ideal is double the point for adj only. To do these. I implement postaging and see which word is nouns, adj, verb… Here is the list nouns that I extracted:`

wine\_train\_set <- clean(train$description)  
  
train\_dtm\_tfidf <- DocumentTermMatrix(wine\_train\_set, control = list(weighting = weightTfIdf))  
train\_dtm\_tfidf <- removeSparseTerms(train\_dtm\_tfidf, 0.99)  
  
#create the test set  
wine\_test\_set <- clean(test$description)  
wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) ,weighting = weightTfIdf))  
#wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) , tokenize=NLP\_tokenizer))  
  
#create matrix for training  
wine\_train\_set <<- as.matrix(train\_dtm\_tfidf)  
wine\_test\_set <- as.matrix(wine\_test\_set)  
wine\_test\_set <- wine\_test\_set[,Terms(train\_dtm\_tfidf)]  
#create the test result  
wine\_testing\_result <- test$quality  
  
  
#tagPos  
tagPOS <- function(x, ...) {  
 s <- as.String(x)  
 word\_token\_annotator <- Maxent\_Word\_Token\_Annotator()  
 a2 <- Annotation(1L, "sentence", 1L, nchar(s))  
 a2 <- NLP::annotate(s, word\_token\_annotator, a2)  
 a3 <- NLP::annotate(s, Maxent\_POS\_Tag\_Annotator(), a2)  
 a3w <- a3[a3$type == "word"]  
 POStags <- unlist(lapply(a3w$features, `[[`, "POS"))  
 POStagged <- paste(sprintf("%s/%s", s[a3w], POStags), collapse = " ")  
 list(POStagged = POStagged, POStags = POStags)  
}  
  
#extract nouns and adj  
tag <- tagPOS(Terms(train\_dtm\_tfidf))  
tag <- tag$POStags  
noun\_id <- which( tag=="NN")  
nouns <- colnames(wine\_train\_set)[noun\_id]  
adj\_id <- which( tag=="JJ")  
adj <- colnames(wine\_train\_set)[adj\_id]

nouns

Here is the adj list:

adj

column\_id <- c()  
  
#multify for noun  
for (i in 1:dim(wine\_train\_set)[2]) {  
 check <- colnames(wine\_train\_set)[i] %in% nouns  
 if(check)  
 {  
 column\_id <- c(column\_id, i)  
 }  
}  
  
wine\_train\_set[,column\_id] <- wine\_train\_set[,column\_id]\*2  
wine\_test\_set[,column\_id] <- wine\_test\_set[,column\_id]\*2  
  
  
wine\_train\_set[1,]

## accent acid across add   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## africa aftertast age alcohol   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## almond almost along alongsid   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## alreadi alsac also although   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## ampl anis anoth appeal   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## appel appl approach apricot   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## argentina aroma aromat around   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## astring attract australia austria   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## back bake balanc barolo   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## barrel beauti berri best   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## better big bit bitter   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## black blackberri blanc blend   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## blossom blue blueberri boast   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## bodi bold bordeaux bordeauxstyl   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## bottl bouquet bright bring   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## brisk brunello brut burgundi   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## burst butter cab cabernet   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## california can candi caramel   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## carri cassi cedar cellar   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## central champagn char charact   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## chardonnay cherri chewi chianti   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## chile chocol chunki cinnamon   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## citrus citrusi classic classico   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## clean close clove coast   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## cocoa coffe cola color   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## columbia combin come complex   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## concentr cool core counti   
## 0.20117624 0.00000000 0.00000000 0.00000000   
## cranberri creami creek crisp   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## cru crush ctes currant   
## 0.00000000 0.55958348 0.00000000 0.00000000   
## cut cuve dark deep   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## del delic delici deliv   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## dens depth despit develop   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## doesnt domin douro dri   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## drink dusti earth earthi   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## easi edg eleg element   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## end enjoy enough espresso   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## estat even excel exot   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## express extra extract famili   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## featur feel ferment fill   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## find fine finger finish   
## 0.00000000 0.11204771 0.32601063 0.04444383   
## firm first flavor fleshi   
## 0.00000000 0.00000000 0.05268637 0.00000000   
## floral flower focus follow   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## food foothil forest forward   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## fragrant frame franc french   
## 0.00000000 0.29776045 0.00000000 0.00000000   
## fresh front fruit fruiti   
## 0.07068905 0.00000000 0.00000000 0.00000000   
## full fullbodi gamay generous   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## gentl germani get give   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## glass good grand grape   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## grapefruit great green grenach   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## grill grip gris grner   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## grown hard heat heavi   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## herb herbal here high   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## hill hint hold honey   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## immedi impress includ integr   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## intens interest itali jam   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## jammi juic juici just   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## keep lack lake last   
## 0.00000000 0.00000000 0.61103055 0.00000000   
## layer lead leaf lean   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## least leather leav lemon   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## lemoni lend length les   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## licoric lift light like   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## lime linger littl live   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## load loir long lot   
## 0.00000000 0.00000000 0.11491235 0.00000000   
## love lush made make   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## malbec mango mani mark   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## matur meat meati medium   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## mediumbodi melon mendoza merlot   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## midpal mild miner mint   
## 0.00000000 0.00000000 0.19314509 0.00000000   
## mix mocha moder montalcino   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## month mountain mouth mouthfeel   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## much napa natur nebbiolo   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## nectarin need new next   
## 0.00000000 0.00000000 0.10399031 0.00000000   
## nice noir north northeastern   
## 0.00000000 0.33136440 0.00000000 0.00000000   
## northern nose note now   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## nuanc nut oak oaki   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## offer old oliv one   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## open opul orang oregon   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## overal pack pair palat   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## paso peach pear peel   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## pepper pepperi perfect perfum   
## 0.00000000 0.00000000 0.00000000 0.26998052   
## persist petit pie piedmont   
## 0.16815139 0.00000000 0.00000000 0.00000000   
## pineappl pink pinot play   
## 0.00000000 0.00000000 0.29631649 0.00000000   
## pleasant plenti plum plump   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## plush polish pomegran portug   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## portugues potenti power premier   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## pretti price produc provenc   
## 0.14222026 0.00000000 0.00000000 0.00000000   
## provid provinc prune pure   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## qualiti quit raci raisin   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## ranch raspberri rather readi   
## 0.00000000 0.19018310 0.00000000 0.00000000   
## red refresh region remain   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## reserv reserva reveal rhne   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## rhnestyl rich riesl right   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## rioja ripe riserva river   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## roast robl ros rose   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## round russian rustic sage   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## sangioves santa sardinia sauvignon   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## savori scent seem select   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## sens set sharp short   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## show sicili side sierra   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## silki simpl sip sirah   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## skin slight smell smoke   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## smoki smooth soft soften   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## soil solid somewhat sonoma   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## soon sour sourc south   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## southern southwest spain sparkl   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## spice spici start still   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## stone straightforward strawberri streak   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## strong structur style subtl   
## 0.00000000 0.00000000 0.00000000 0.14673065   
## sugar suggest superior suppl   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## support sweet syrah take   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## tangerin tangi tannic tannin   
## 0.00000000 0.00000000 0.00000000 0.10718230   
## tart tast tea tempranillo   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## textur that there thick   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## though tight time toast   
## 0.00000000 0.00000000 0.00000000 0.20854769   
## toasti tobacco togeth tomato   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## tone top toscana touch   
## 0.28126048 0.00000000 0.00000000 0.00000000   
## tropic turn tuscani two   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## valley vanilla variet varieti   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## veltlin velveti veneto verdot   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## veri vibrant vine vineyard   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## vintag viognier violet warm   
## 0.00000000 0.00000000 0.29381041 0.00000000   
## washington way weight well   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## wet whiff white wild   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## will willamett wine winemak   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## wineri without wonder wood   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## wrap year yellow yet   
## 0.00000000 0.00000000 0.00000000 0.00000000   
## york young youth zealand   
## 0.30057652 0.00000000 0.00000000 0.00000000   
## zest zesti zinfandel   
## 0.00000000 0.00000000 0.00000000

here is the resut:

train\_svmLinear\_model <- train(x= wine\_train\_set, y=train$quality , method = 'svmLinear3')  
train\_svmLinear\_model  
model\_svmLinear\_result <- predict(train\_svmLinear\_model, newdata = wine\_test\_set)  
  
conf\_svmLinear\_train <- table(model\_svmLinear\_result, wine\_testing\_result)  
names(dimnames(conf\_svmLinear\_train)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_svmLinear\_train)

We can see that accurancy is slightly increse (just 0.3%). FN is slightly decrease. Althought number of FP is increse but overall, I conclude that the model is better.

Now double the point only for adj:

#remove double point for nouns  
wine\_train\_set[,column\_id] <- wine\_train\_set[,column\_id]/2  
wine\_test\_set[,column\_id] <- wine\_test\_set[,column\_id]/2  
  
#multify for adj  
column\_id <- c()  
for (i in 1:dim(wine\_train\_set)[2]) {  
 check <- colnames(wine\_train\_set)[i] %in% adj  
 if(check)  
 {  
 column\_id <- c(column\_id, i)  
 }  
}  
  
wine\_train\_set[,column\_id] <- wine\_train\_set[,column\_id]\*2  
wine\_test\_set[,column\_id] <- wine\_test\_set[,column\_id]\*2  
  
wine\_train\_set[1,]

And here is the result:

train\_svmLinear\_model <- train(x= wine\_train\_set, y=train$quality , method = 'svmLinear3')  
train\_svmLinear\_model  
model\_svmLinear\_result <- predict(train\_svmLinear\_model, newdata = wine\_test\_set)  
  
conf\_svmLinear\_train <- table(model\_svmLinear\_result, wine\_testing\_result)  
names(dimnames(conf\_svmLinear\_train)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_svmLinear\_train)

The accurancy is just increase 0.2%. But the FN is decrese 29 units, quite impresive. TP for excellent class is also higher. Although FN is also higher. I conclude that this is the best model. To explain why double the point for adj is better than double the point for nouns. When review the dataset. I see there are two resons: Firstly, mosst important nouns is ingredient, country, region.. They can use the same ingerdient, but the experts may have some teachique to make it become excellent wine. So, nouns is not so helpful. But adjactive, which means the tease or the properties of the wines will show clearly the different between good and excellent wine. Secondly, because I steam when process data. Many original words is steamed to nouns but it is not the nouns in the original paragrahp. (we can see that the number of nouns is higher than number of adj) Which leads to the method is not useful as expected.

Here is the final result if I try with 100% of data:

wine\_train\_set\_full <- clean(train\_full$description)  
  
train\_dtm\_tfidf\_full <- DocumentTermMatrix(wine\_train\_set\_full, control = list(weighting = weightTfIdf))  
train\_dtm\_tfidf\_full <- removeSparseTerms(train\_dtm\_tfidf\_full, 0.99)  
  
#create the test set  
wine\_test\_set\_full <- clean(test\_full$description)  
wine\_test\_set\_full <- DocumentTermMatrix(wine\_test\_set\_full, control = list(dictionary = Terms(train\_dtm\_tfidf\_full) ,weighting = weightTfIdf))  
#wine\_test\_set <- DocumentTermMatrix(wine\_test\_set, control = list(dictionary = Terms(train\_dtm\_tfidf) , tokenize=NLP\_tokenizer))  
  
#create matrix for training  
wine\_train\_set\_full <<- as.matrix(train\_dtm\_tfidf\_full)  
wine\_test\_set\_full <- as.matrix(wine\_test\_set\_full)  
wine\_test\_set\_full <- wine\_test\_set\_full[,Terms(train\_dtm\_tfidf\_full)]  
#create the test result  
wine\_testing\_result\_full <- test\_full$quality  
  
tag\_full <- tagPOS(Terms(train\_dtm\_tfidf\_full))  
tag\_full <- tag\_full$POStags  
adj\_id\_full <- which( tag\_full=="JJ")  
adj\_full <- colnames(wine\_train\_set)[adj\_id]  
  
column\_id\_full <- c()  
  
#multify for noun  
for (i in 1:dim(wine\_train\_set\_full)[2]) {  
 check <- colnames(wine\_train\_set\_full)[i] %in% adj\_full  
 if(check)  
 {  
 column\_id\_full <- c(column\_id\_full, i)  
 }  
}  
  
wine\_train\_set[,column\_id\_full] <- wine\_train\_set[,column\_id\_full]\*2  
wine\_test\_set[,column\_id\_full] <- wine\_test\_set[,column\_id\_full]\*2  
  
  
train\_svmLinear\_model\_final1 <- train(x= wine\_train\_set\_full, y=train\_full$quality , method = 'svmLinear3')  
train\_svmLinear\_model\_final1  
model\_svmLinear\_result\_final1 <- predict(train\_svmLinear\_model\_final1, newdata = wine\_test\_set\_full)  
  
  
conf\_svmLinear\_train\_final1 <- table(model\_svmLinear\_result\_final1, wine\_testing\_result\_full)  
names(dimnames(conf\_svmLinear\_train\_final1)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_svmLinear\_train\_final1)

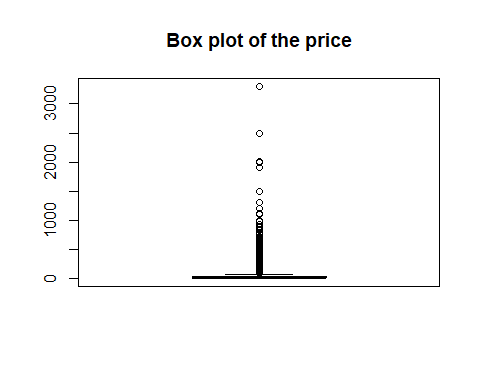
Result….

# second aspect: high value wine

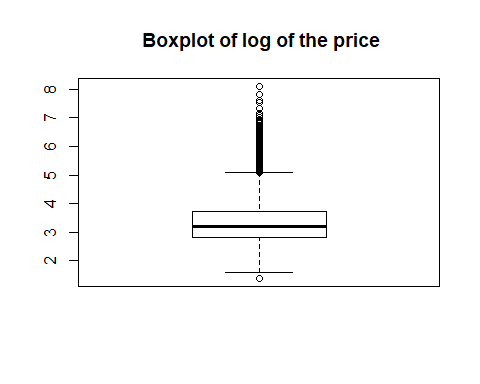
## The problem

In most of time, the price of the bottle can reflect the quatity of the wine. For example, If you want to buy an excellent wine, choose the 1000$ bottle. That method should work. But, most of us don’t have the financial condition for buying like this. The point is if we have a limit amount of money, or we just don’t want to pay too much for a bottle, how can we choose? It’s my idea for the second problem. Let’s take a look at the price of our data:

boxplot(wine$price, main = "Box plot of the price")



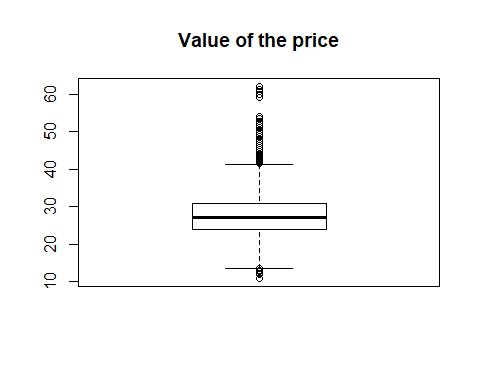
boxplot(log(wine$price), main = " Boxplot of log of the price")

 Because the price range quite big, have a lot of outlier and high value of SD. otherwise, the log of the price look better and quite similar with the point.

## Aim

I define the value = point/log(price). Here is the value.

boxplot(wine$value, main = "Value of the price")



The median number for value is ~27. So the bottle with higher value than 27 will have “high” benefit. The rest have “medium” benefit. This way of define class will make all data become interesting. For example. there is a bottle with just 4$ but have 80 point. It become the bottle with highest benefit. In constrast. many bottle with the price higher than 1000 dollar. Of couse they are excellent wine. But is just have medium benefit because it’s too expensive.

I do the same preprocessing for the text. Then continue compare models with sample 20% of the data. Here is the result.

#remove for adj  
for (i in 1:dim(wine\_train\_set)[2]) {  
 check <- colnames(wine\_train\_set)[i] %in% nouns  
 if(check)  
 {  
 column\_id <- c(column\_id, i)  
 }  
}  
  
wine\_train\_set[,column\_id] <- wine\_train\_set[,column\_id]/2  
wine\_test\_set[,column\_id] <- wine\_test\_set[,column\_id]/2  
  
  
wine\_testing\_result <- test$benefit

train\_nb\_model2 <- train(x= wine\_train\_set, y=train$benefit , method = 'naive\_bayes')  
train\_nb\_model2  
model\_nb\_result2 <- predict(train\_nb\_model2, newdata = wine\_test\_set)  
  
conf\_nb\_train2 <- table(model\_nb\_result2, wine\_testing\_result)  
names(dimnames(conf\_nb\_train2)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_nb\_train2)

train\_svmLinear\_model2 <- train(x= wine\_train\_set, y=train$benefit , method = 'svmLinear3')  
train\_svmLinear\_model2

model\_svmLinear\_result2 <- predict(train\_svmLinear\_model2, newdata = wine\_test\_set)  
  
conf\_svmLinear\_train2 <- table(model\_svmLinear\_result2, wine\_testing\_result)  
names(dimnames(conf\_svmLinear\_train2)) <- c("Predicted class", "Actual class")  
confusionMatrix(conf\_svmLinear\_train2)

As we can see, SVM Linear is better than Nb in both accurancy and how the bottle is classify. But in this case, Pos tagging don’t show any clearly effect either with double the point for nouns or adjective

As I said before, The way of classify here is quite strage. I still don’t too much ideal about how the description should be for each class.

final model for “high” - “medium” value

# Futher discusssion

## Suggession for choosing wine

In this section, I’ll extract the top 20 influences of each model and present in a word cloud. The big if the word is related to the points of important Here is the result from the SVM Linear for “good” - “excellent” wine model.

Here is the result from the SVM Linear for “good” - “excellent” wine model.

## Future work:

* Here is some points that can improve from the project:
* Check again the SVMRBF kenels with a bigger data set and more powerful computer. In my experiments, SVM RBF is better than SVM Linear in most of cases.
* Take advices of wine specilist for improve the model. In my opinion, we should fully understand the dataset before thinking about machine learning. In this case. Because my knowleadge about wine is limited. I often don’t clearly understand about a bottle after read the description. It leads to the ideal about improve the model is limited.

# Conclusion

The issues raised at the begining are solved with a good result. We firuge out that the description and other information printed in a bottle can bring us some ideal about how good the bottle is or how high level of benefit it have. We also have some idea about how can we choose a bottle. In the NLP and ML aspect. SVM linear with double tf-idf point for adjtive is the best model for predict the quality with aa% accurancy and bb% for FN. For predict the benefit of a bottle, SVM linear is a acceptable model, with accurancy 77.09% and FN cc%

# References

# Apendix

This report is writen ussing Rmarkdown. Here is the code of this project: