Naive Bayes to predict ratings based on Yelp restaurant reviews

Nisha Selvarajan

3 October 2020

Contents

Objectives:

Does the Rating and Review match in Yelp? On famous websites like Amazon and Yelp, many products and businesses receive tens or hundreds of reviews, making it impossible for readers to read all of them.Generally, readers prefer to look at the star ratings only and ignore the text. However, the relationship between the text and the rating is not obvious. In particular, several questions may be asked: why exactly did this reviewer give the restaurant 3/5 stars? In addition to the quality of food, variety, size and service time, what other features of the restaurant did the user implicitly consider, and what was the relative importance given to each of them? How does this relationship change if we consider a different user's rating and text review? The process of predicting this relationship for a generic user is called Review Rating Prediction

The main challenge which we will solve is building a good predictor which effectivelys extract useful features of the product from the text reviews and then quantify their relative importance with respect to the rating.

Data Description

- 33K Rows, with 17 columns. You can download data on the link https://www.kaggle.com/shikhar42/yelps-dataset

```
library(knitr)
library(kableExtra)
knitr::opts_chunk$set(message = FALSE)
knitr::opts_chunk$set(warning = FALSE)


df <- data.frame(Names = c("business_id","name","address",
                           "postal_code","latitude","longitude",
                           "stars","review_count","is_open",
                           "review_id","user_id"),
                 Description = c("ID related to each business","Name of the business",
                 "Street Adress of the business", "zip code of the business",
                 "Latitude of the business", "Longitude of the business" ,
                 "Rating given by user to the business", "Total number of
                 reviews a user had posted at the time of data collection",
                 "Restaraunt open or closed","Unique Review Id","User Id of the reviewer"
                ))
```

```
kbl(df) %>%
  kable_paper(full_width = F) %>%
  column_spec(2, width = "30em")
```

Names	Description
business_id	ID related to each business
name	Name of the business
address	Street Adress of the business
postal_code	zip code of the business
latitude	Latitude of the business
longitude	Longitude of the business
stars	Rating given by user to the business
review_count	Total number of reviews a user had posted at the time of data collection
is_open	Restaraunt open or closed
review_id	Unique Review Id
user_id	User Id of the reviewer

Using Naive Bayes to validate the review and ratings

- ***Step 1: import dataset***

```
library(class)
library(knitr)
library(kableExtra)
library(caret)
library(tidyverse)
library(tokenizers)
library(tidytext)
library(wordcloud)
library(tm)
library(dplyr)
library(caret)
library(naivebayes)
library(wordcloud)
yelpdataset=read.csv(file = "/Users/nselvarajan/Desktop/test/archive/cleaned.csv", sep = ",")
yelpdataset <- data.frame(yelpdataset, stringsAsFactors = FALSE)
head(yelpdataset)
```

```
##              business_id         name                address postal_code
## 1 rDMptJYWtnMhpQu_rRXHng "McDonald's" "719 E Thunderbird Rd"       85022
## 2 rDMptJYWtnMhpQu_rRXHng "McDonald's" "719 E Thunderbird Rd"       85022
## 3 rDMptJYWtnMhpQu_rRXHng "McDonald's" "719 E Thunderbird Rd"       85022
## 4 rDMptJYWtnMhpQu_rRXHng "McDonald's" "719 E Thunderbird Rd"       85022
## 5 rDMptJYWtnMhpQu_rRXHng "McDonald's" "719 E Thunderbird Rd"       85022
## 6 rDMptJYWtnMhpQu_rRXHng "McDonald's" "719 E Thunderbird Rd"       85022
##   latitude longitude stars_res review_count is_open              review_id
## 1 33.60707 -112.0644         1           10       1 bABGON0ehmb7MBJrI0217Q
## 2 33.60707 -112.0644         1           10       1 zn7bEYAVzwWSJdSd2a4zoQ
## 3 33.60707 -112.0644         1           10       1 ONnRwv_KOLRyKyk72SzTHg
## 4 33.60707 -112.0644         1           10       1 wlcWp7STNYOCcnpap2_Nzw
## 5 33.60707 -112.0644         1           10       1 OBsbVLK2dLyT55Nw-omXRA
## 6 33.60707 -112.0644         1           10       1 nSq8oldCoOHxhvfvc2D7SQ
##                  user_id stars    date
```

```
## 1 Ck73f1qtZbu68F_vjzsBrQ     1 2/25/16
## 2 u0JoB0Vm1ZhwF8nysxPnfg     2  6/6/11
## 3 F95NFEFwuwA__SIRt9IJNA     1 11/5/15
## 4 uHZxYHgjxhXY7PS6g2rFsA     1 6/17/12
## 5 Akt0l1UBaVa1Qxi8Ogdv4Q     1 8/12/11
## 6 2gWCW1oEuyhaxrlTTghvtQ     1 8/27/17
##
## 1 The speed of delivery of my food order was terrible.  It took 10 minutes from the time of my order
## 2
## 3
## 4
## 5
## 6
##   useful funny cool
## 1      3     0    0
## 2      6     7    4
## 3      2     0    0
## 4      1     0    0
## 5      0     0    0
## 6      0     0    0
```

- ***Step 2: Clean the Data***
  - Create a outcome variable which is a true or false indicator specifying if the sentiment corresponding to the review is positive or not.

```
yelpdataset$positive = as.factor(yelpdataset$stars > 3)
```

- ***Step 3: Features and Preprocessing***
  - Load the data into a Corpus (a collection of documents) which is the main data structure used by tm.
  - Review texts were cleaned by tm package which provides several function to clean the text via the tm_map() function.
  - Cleaning proces include removing format, punctuation and extra whitespace.All characters from the dataset are lowercase, so there is no need to preprocess uppercase letters. Word stemming was achieved using Porter stemming algorithm, which erased word suffixes to retrieve the root or stem. Stopwords, that is, words with no information value but appear too common in a language, were also removed according to a list from nltk.corpus.

```
myCorpus <- Corpus(VectorSource(yelpdataset$text))
corpus <- tm_map(myCorpus,removeNumbers)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, tolower)
corpus <- tm_map(corpus, stemDocument, language = 'english')
corpus <- tm_map(corpus, removeWords, stopwords('english'))
corpus <- tm_map(corpus, stripWhitespace)
```

- ***Step 4: Making a document-term matrix***
  - The document-term matrix is used when you want to have each document represented as a row.
  - Bag of words is a way to count terms, n-grams, across a collection of documents.
  - Create dataframe from cleaned corpus

```
bag_of_words <- DocumentTermMatrix(corpus)
inspect(bag_of_words)
```

```
## <<DocumentTermMatrix (documents: 331400, terms: 158826)>>
## Non-/sparse entries: 14800215/52620136185
```

```
## Sparsity           : 100%
## Maximal term length: 256
## Weighting          : term frequency (tf)
## Sample             :
##         Terms
## Docs      food good great just like order place servic time veri
##   122288    4    1     1    2    6     1     1      0    0    1
##   126026    2    2     2    4    7     1     1      0    4    2
##   195082    1    3     2    1    3     2     4      0    4    2
##   280215    1    3     2    0    2     1     0      0    0    1
##   31177     3    6     3    2    2     7     6      6    3    2
##   330939    1    1     3    1    1     1     0      2    0    2
##   50554     2    1     2    2    4     3     3      0    4    0
##   75719     0    4     2    1    7     2     0      0    1    2
##   80183     0    1     0    5    8     1     1      1    3    1
##   91714     6    4     0   16    4     9     6      2    6    0
```

```r
dataframe<-data.frame(text=unlist(sapply(corpus, `[`)), stringsAsFactors=F)
yelpdataset$text <- dataframe$text
```

- ***Step 5: Build Word Cloud***
  - Word cloud is a fun way to visualize popular words/phrases in group of text.
  - This function takes a single parameter of review text and builds word clouds for words occuring with the highest frequencies in reviews for these restaurants

```r
y<-head(yelpdataset,100)
library(wordcloud)
wordcloud(y$text)
```

- ***Step 6: Build Word Cloud For 5 Star Reviews***
  - Build word cloud for 5 star ratings.

```
rating5 <- subset(yelpdataset, stars == "5")  ##Filtering data for 5 star reviews
myCorpusRating5 <- Corpus(VectorSource(rating5$text))
myCorpusRating5 <- tm_map(myCorpusRating5,removeNumbers)
myCorpusRating5 <- tm_map(myCorpusRating5, removePunctuation)
myCorpusRating5 <- tm_map(myCorpusRating5, tolower)
myCorpusRating5 <- tm_map(myCorpusRating5, stemDocument, language = 'english')
myCorpusRating5 <- tm_map(myCorpusRating5, removeWords, stopwords('english'))
myCorpusRating5 <- tm_map(myCorpusRating5, stripWhitespace)
bag_of_words_rating_5 <- DocumentTermMatrix(myCorpusRating5)
##creating DTM to get frequencies
inspect(bag_of_words_rating_5)
```

```
## <<DocumentTermMatrix (documents: 140341, terms: 77011)>>
## Non-/sparse entries: 5285796/10802514955
## Sparsity           : 100%
## Maximal term length: 180
## Weighting          : term frequency (tf)
## Sample             :
##         Terms
## Docs     food friend good great love order place servic time veri
##   106225    2      0    4     1    4     4     5      0    5    7
##   117499    2      0    1     0    0     2     3      0    2    1
##   134018    1      1    2     3    1     2     3      0    2    1
##   140126    1      1    1     3    2     1     0      2    0    2
##   32730     0      0    4     2    3     2     0      0    1    2
##   42102     3      2    2     4    4     2     7      1    2    0
##   51703     4      0    1     1    1     1     1      0    0    1
##   55581     1      1    6     2    2     1     1      0    0    1
##   90524     0      0    3     2    0     1     4      2    4    1
##   97647     4      0    0     4    5     2     7      0    2    8
```

```
dataframeRating5<-data.frame(text=unlist(sapply(myCorpusRating5,
`[`)), stringsAsFactors=F) ##creating data fram from matrix
yFiveStar<-head(dataframeRating5,100)
# word cloud visualization
wordcloud(yFiveStar$text)
```

- ***Step 7: Model Training and Testing***
  - I used 25% to test data and 75% to data train.
  - After obtaining training and testing data sets, then we will create a separate data frame which has values to be compared with actual final values

```
dataset_train <- yelpdataset[1:24000,]  ###dividing data into training and test set
dataset_test <- yelpdataset[24000:331400,]
 ##creating corpus for training
myCorpus_model_train <- Corpus(VectorSource(dataset_train$text))
 ##since this data was already cleaned before, we can straigtaway move to DTM
dtm_train <- DocumentTermMatrix(myCorpus_model_train)
dtm_train <- removeSparseTerms(dtm_train,0.95)
##creating corpus for test
myCorpus_model_test <- Corpus(VectorSource(dataset_test$text))
 ##since this data was already cleaned before, we can straigtaway move to DTM
dtm_test <- DocumentTermMatrix(myCorpus_model_test)
dtm_test <- removeSparseTerms(dtm_test,0.95)
```

- ***Step 8: Making predictions***
- We build Naive Bayes by using training & test data sets.
- We apply Laplace smoothing , which is a technique for smoothing categorical data.
- A small-sample correction, or pseudo-count, will be incorporated in every probability estimate. Consequently, no probability will be zero. this is a way of regularizing Naive Bayes, and when the pseudo-count is zero, it is called Laplace smoothing.

```
model <- naive_bayes(as.data.frame(as.matrix(dtm_train)), dataset_train$positive, laplace = 1)
model
```

```
##
```

```
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.default(x = as.data.frame(as.matrix(dtm_train)),
##     y = dataset_train$positive, laplace = 1)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 1
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##
##     FALSE      TRUE
## 0.2692917 0.7307083
##
## ---------------------------------------------------------------------------------
##
##  Tables:
##
## ---------------------------------------------------------------------------------
##  ::: check (Gaussian)
## ---------------------------------------------------------------------------------
##
## check       FALSE       TRUE
##   mean 0.10165558 0.05628101
##   sd   0.40198599 0.26564318
##
## ---------------------------------------------------------------------------------
##  ::: disappoint (Gaussian)
## ---------------------------------------------------------------------------------
##
## disappoint      FALSE       TRUE
##       mean 0.14265821 0.05730741
##       sd   0.39347611 0.24275615
##
## ---------------------------------------------------------------------------------
##  ::: food (Gaussian)
## ---------------------------------------------------------------------------------
##
## food       FALSE      TRUE
##   mean 0.9413585 0.6349433
##   sd   1.1963510 0.8701281
##
## ---------------------------------------------------------------------------------
##  ::: great (Gaussian)
## ---------------------------------------------------------------------------------
##
## great      FALSE      TRUE
##   mean 0.2846975 0.5872156
##   sd   0.6195211 0.8539856
##
## ---------------------------------------------------------------------------------
```

```
##  ::: high (Gaussian)
## ---------------------------------------------------------------------------------
##
## high        FALSE       TRUE
##   mean 0.07086492 0.07692308
##   sd   0.28025496 0.28708026
##
## ---------------------------------------------------------------------------------
##
## # ... and 176 more tables
##
## ---------------------------------------------------------------------------------
```

Interpretation of the results and prediction accuracy achieved

- ***Evaluate the model performance using confusionMatrix***
- The accuracy of our model on the testing set is 72%.
- We can visualise the model's performance using a confusion matrix.
- We can evaluvate the accuracy, precision and recall on the training and validation sets to evaluate the performance of naive bayes algorithm.

```r
model_predict <- predict(model, as.data.frame(as.matrix(dtm_test)))
confusionMatrix(model_predict, dataset_test$positive)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  FALSE   TRUE
##      FALSE  46645  32147
##      TRUE   51353 177256
##
##                Accuracy : 0.7284
##                  95% CI : (0.7268, 0.7299)
##     No Information Rate : 0.6812
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3402
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4760
##             Specificity : 0.8465
##          Pos Pred Value : 0.5920
##          Neg Pred Value : 0.7754
##              Prevalence : 0.3188
##          Detection Rate : 0.1517
##    Detection Prevalence : 0.2563
##       Balanced Accuracy : 0.6612
##
##        'Positive' Class : FALSE
##
```

- ***Evaluate the model performance using CrossTable***

```r
library(gmodels)
CrossTable(model_predict, dataset_test$positive,
```

```r
        prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
        dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  307401
##
##
##              | actual
##    predicted |     FALSE |      TRUE | Row Total |
## -------------|-----------|-----------|-----------|
##        FALSE |     46645 |     32147 |     78792 |
##              |     0.476 |     0.154 |           |
## -------------|-----------|-----------|-----------|
##         TRUE |     51353 |    177256 |    228609 |
##              |     0.524 |     0.846 |           |
## -------------|-----------|-----------|-----------|
## Column Total |     97998 |    209403 |    307401 |
##              |     0.319 |     0.681 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Overall insights obtained from the implemented project

- Overall accuracy of the model is 72%.It is safe to assume that naive bayes models can be trained on to find the rating of the restaurant based on the reviews.
- Sensitivity for finding ratings is 0.4760.
- Specificity for finding ratings is 0.8465.
- Since the dataset was clean, and reviews are equally distributed between test & training set, adding laplace smoothing factor did not make much difference in the accuracy.