Garg_Bheeni_Stat6620_Project2

Bheeni Garg

June 10, 2016

One of the job descriptions I found online—

Responsibilities:

-Analyze and model structured data and implement algorithms to support analysis using advanced statistical and mathematical methods from statistics, machine learning, data mining, econometrics, and operations research

-Perform Statistical Natural Language Processing to mine unstructured data, using methods such as document clustering, topic analysis, named entity recognition, document classification, and **sentiment analysis**

So, I chose to do the following project——

Project – Sentiment Analysis of Yelp Ratings using Naive Bayes Classifier——

Introduction:

Yelp, founded in 2004, is a multinational corporation that publishes crowd-sourced online reviews on local businesses.

As of 2014, Yelp.com had 57 million reviews and 132 million monthly visitors [1]. A portion of their large dataset is available on the Yelp Dataset Challenge homepage, which includes data on 42,153 businesses, 252,898 users, and 1,125,458 reviews from the cities of Phoenix, Las Vegas, Madison, Waterloo, and Edinburgh [2]. For businesses, the dataset includes business name, neighborhood, city, latitude and longitude, average review rating, number of reviews, and categories such as "good for lunch". The dataset also includes review text and rating.

In this project, I have attempted to build a classifier to classify reviews as either 5-star or 1-star using only the review text! This may further be used to identify potential factors that may affect business performance on Yelp and then predict future ratings based on the identified important features. Sentiments, supposedly, have the highest predictive power and hence need to classified accurately.

Step 1: Collecting Data

The data used for the project is taken from the Kaggle Competition page link, Yelp Business Rating Prediction.

The dataset consists of 10,000 observations with 10 features.

Step 2: Exploring and preparing data——

```r
yelp <- read.csv("yelp.csv")
str(yelp)
```

```
## 'data.frame':    10000 obs. of  10 variables:
##  $ business_id: Factor w/ 4174 levels "_-9pMxBWtG_x814rHWBasg",..: 774 4138 565 2 566 135 4134 1773 
##  $ date       : Factor w/ 1995 levels "2005-04-18","2005-07-03",..: 1286 1468 1790 1045 1629 225 943
```

```
##  $ review_id  : Factor w/ 10000 levels "__esH_kgJZeS8k3i6HaG7Q",..: 3754 4520 4479 3792 632 5645 7375
##  $ stars      : int  5 5 4 5 5 4 5 4 4 5 ...
##  $ text       : Factor w/ 9998 levels "- the location is excellent\n- the food is mediocre, and milde
##  $ type       : Factor w/ 1 level "review": 1 1 1 1 1 1 1 1 1 1 ...
##  $ user_id    : Factor w/ 6403 levels "__FXEOrWIjXMOElz2pGlBQ",..: 4693 215 244 5373 5568 4934 5658 3
##  $ cool       : int  2 0 0 1 0 4 7 0 0 0 ...
##  $ useful     : int  5 0 1 2 0 3 7 1 0 1 ...
##  $ funny      : int  0 0 0 0 0 1 4 0 0 0 ...
```

```r
df1 <- subset(yelp, stars == 1 | stars == 5)
df1[10:20, ]
```

```
##                business_id       date              review_id stars
## 18 0510Re68mOy9dU490JTKCg 2010-05-03 j4SIzrIy0WrmW4yr4--Khg     5
## 22 tdcjXyFLMKAsvRhURNOkCg 2011-06-28 LmuKVFh03Uz318VKnUWrxA     5
## 23 eFA9dqXT5EA_TrMgbo03QQ 2011-07-13 CQYc8hgKxV4enApDkx0IhA     5
## 24 IJ0o6b8bJFAbG6MjGfBebQ 2010-09-05 Dx9sfFU6Zn0GYOckijom-g     1
## 25 JhupPnWfNlMJivnWB5druA 2011-05-22 cFtQnKzn2VDpBedy_TxlvA     5
## 27 qjmCVYkwP-HDa35jwYucbQ 2013-01-03 kZ4TzrVX6qeF0OvrVTGVEw     5
## 31 V1nEpIRmEa1768oj_tuxeQ 2011-05-09 dtpJXC5p_sdWDLSobluJ3Q     5
## 32 vvA3fbps4F9nGlAEYKk_sA 2012-05-04 S9OVpXat8k5YwWCn6FAgXg     1
## 33 rxQ2PIjhAx6dgAqUalf99Q 2012-09-09 -v-shjbxoj7hpU62yn6vag     5
## 36 o1GIYYZJjM6nM03fQs_uEQ 2011-11-30 ApKbwpYJdnhhgP4NbjQw2Q     1
## 47 aRkYtXfmEKYG-eTDf_qUsw 2009-04-04 Ckk1Cne1GHwzmJfo7M4r2w     5
##
## 18
## 22
## 23
## 24
## 25
## 27
## 31
## 32
## 33 Never having dealt with a Discount Tire in Phoenix before (only in Texas, and their service has be
## 36
## 47
##      type                user_id cool useful funny
## 18 review u1KWcbPMvXFEEYkZZ0Yktg    0      0     0
## 22 review YN3ZLOdg8kpnfbVcIhuEZA    1      1     2
## 23 review 6lg55RIP23VhjYEBXJ8Njw    0      0     0
## 24 review zRlQEDYd_HKp0VS3hnAffA    0      1     1
## 25 review 13xj6FSvYO0rZVRv5XZp4w    0      1     0
## 27 review fpItLlgimq0nRltWOkuJJw    0      0     0
## 31 review bCKjygWJZOQHCOzootbvow    0      2     0
## 32 review 8AMn6644NmBf96xGO3w6OA    0      1     0
## 33 review HLbhD2OyiMCUDRR4c1iXaw    0      0     0
## 36 review iwUN95LIaEr75TZE_JC6bg    0      4     3
## 47 review IUWjTmXc3wLVaMHz33inaA    2      1     1
```

```r
yelp_new <- as.data.frame(df1[, c("stars", "text")])

# examine the structure of yelp data
str(yelp_new)
```

```
## 'data.frame':    4086 obs. of  2 variables:
##  $ stars: int  5 5 5 5 5 5 5 5 5 5 ...
##  $ text : Factor w/ 9998 levels "- the location is excellent\n- the food is mediocre, and milder than
```

```
# convert stars 1, 5 to factor.
yelp_new$stars <- factor(yelp_new$stars)

# convert text to character
yelp_new$text <- as.character(yelp_new$text)

# examine the type variable more carefully
str(yelp_new$stars)
```

```
##  Factor w/ 2 levels "1","5": 2 2 2 2 2 2 2 2 2 2 ...
```

```
str(yelp_new$text)
```

```
##  chr [1:4086] "My wife took me here on my birthday for breakfast and it was excellent.  The weather w
```

```
str(yelp_new)
```

```
## 'data.frame':    4086 obs. of  2 variables:
##  $ stars: Factor w/ 2 levels "1","5": 2 2 2 2 2 2 2 2 2 2 ...
##  $ text : chr  "My wife took me here on my birthday for breakfast and it was excellent.  The weather
```

```
table(yelp_new$stars)
```

```
##
##    1    5
##  749 3337
```

```
# build a corpus using the text mining (tm) package
library(tm)
yelp_corpus <- VCorpus(VectorSource(yelp_new$text))

# examine the yelp_new corpus
print(yelp_corpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 4086
```

```
inspect(yelp_corpus[1:2])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 2
##
## [[1]]
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 889
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 1345
```

```r
as.character(yelp_corpus[[1]])
```

```
## [1] "My wife took me here on my birthday for breakfast and it was excellent.  The weather was perfec
```

```r
lapply(yelp_corpus[1:2], as.character)
```

```
## $`1`
## [1] "My wife took me here on my birthday for breakfast and it was excellent.  The weather was perfec
##
## $`2`
## [1] "I have no idea why some people give bad reviews about this place. It goes to show you, you can p
```

```r
# clean up the corpus using tm_map()
yelp_corpus_clean <- tm_map(yelp_corpus, content_transformer(tolower))

# show the difference between sms_corpus and corpus_clean
as.character(yelp_corpus_clean[[1]])
```

```
## [1] "my wife took me here on my birthday for breakfast and it was excellent.  the weather was perfec
```

```r
yelp_corpus_clean <- tm_map(yelp_corpus_clean, removeNumbers)  # remove numbers
yelp_corpus_clean <- tm_map(yelp_corpus_clean, removeWords, stopwords())  # remove stop words
yelp_corpus_clean <- tm_map(yelp_corpus_clean, removePunctuation)  # remove punctuation

library(SnowballC)
yelp_corpus_clean <- tm_map(yelp_corpus_clean, stemDocument)  # remove word stems
yelp_corpus_clean <- tm_map(yelp_corpus_clean, stripWhitespace)  # eliminate unneeded whitespace

# examine the final clean corpus
lapply(yelp_corpus_clean[1:3], as.character)
```

```
## $`1`
## [1] " wife took birthday breakfast excel weather perfect made sit outsid overlook ground absolut plea
##
## $`2`
## [1] " idea peopl give bad review place goe show can pleas everyon probabl gripe someth fault mani pe
##
## $`3`
## [1] "rosi dakota love chaparr dog park conveni surround lot path desert xeriscap basebal field ballpa
```

```r
# create a document-term sparse matrix
yelp_dtm <- DocumentTermMatrix(yelp_corpus_clean)

# compare the result
str(yelp_dtm)
```

```
## List of 6
##  $ i       : int [1:201421] 1 1 1 1 1 1 1 1 1 1 ...
##  $ j       : int [1:201421] 27 349 351 489 614 836 1164 1174 1250 1311 ...
##  $ v       : num [1:201421] 2 1 1 1 1 1 2 1 1 1 ...
##  $ nrow    : int 4086
##  $ ncol    : int 15041
##  $ dimnames:List of 2
##   ..$ Docs : chr [1:4086] "1" "2" "3" "4" ...
##   ..$ Terms: chr [1:15041] "aaa" "aaaamazing" "aaammmazzing" "aaron" ...
##  - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix"
##  - attr(*, "weighting")= chr [1:2] "term frequency" "tf"
```

```r
# creating training and test datasets
require(caTools)
set.seed(101)

yelp_dtm_train <- yelp_dtm[1:2860, ]
yelp_dtm_test <- yelp_dtm[2861:4086, ]

# also save the labels
yelp_train_labels <- yelp_new[1:2860, ]$stars
yelp_test_labels <- yelp_new[2861:4086, ]$stars

# check that the proportion of ratings is similar
prop.table(table(yelp_train_labels))
```

```
## yelp_train_labels
##         1         5
## 0.1783217 0.8216783
```

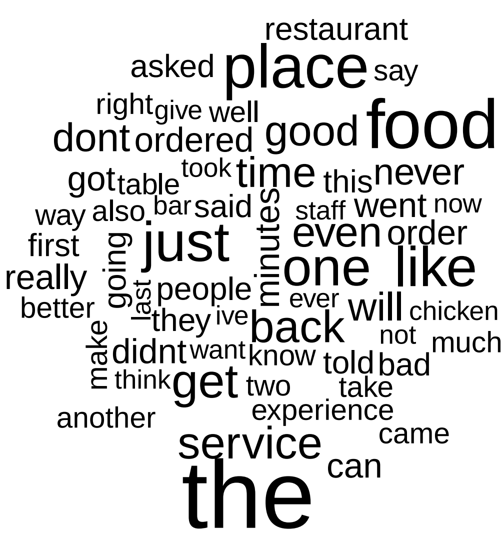
```r
prop.table(table(yelp_test_labels))
```

```
## yelp_test_labels
##         1         5
## 0.1949429 0.8050571
```

```r
# word cloud visualization
library(wordcloud)
wordcloud(yelp_corpus_clean, min.freq = 100, random.order = FALSE)
```


```r
# subset the training data into star 1 and star 5 groups
star1 <- subset(yelp_new, stars == "1")
star5 <- subset(yelp_new, stars == "5")

wordcloud(star1$text, max.words = 60, scale = c(3, 0.5))
```



```r
wordcloud(star5$text, max.words = 60, scale = c(3, 0.5))
```


```r
yelp_dtm_freq_train <- removeSparseTerms(yelp_dtm_train, 0.999)
yelp_dtm_freq_train[1:10, ]
```

```
## <<DocumentTermMatrix (documents: 10, terms: 4377)>>
## Non-/sparse entries: 435/43335
## Sparsity           : 99%
## Maximal term length: 15
## Weighting          : term frequency (tf)
```

```r
# save frequently-appearing terms to a character vector
yelp_freq_words <- findFreqTerms(yelp_dtm_train, 5)
str(yelp_freq_words)
```

```
##  chr [1:3399] "abil" "abl" "absolut" "accent" "accept" ...
```

```r
# create DTMs with only the frequent terms
yelp_dtm_freq_train <- yelp_dtm_train[, yelp_freq_words]
yelp_dtm_freq_test <- yelp_dtm_test[, yelp_freq_words]

# convert counts to a factor
convert_counts <- function(x) {
    x <- ifelse(x > 0, "Yes", "No")
}

# apply() convert_counts() to columns of train/test data
yelp_train <- apply(yelp_dtm_freq_train, MARGIN = 2, convert_counts)
yelp_test <- apply(yelp_dtm_freq_test, MARGIN = 2, convert_counts)
```

Step 3: Training a model on the data —-

```r
library(e1071)
rating_classifier <- naiveBayes(yelp_train, yelp_train_labels)
```

Step 4: Evaluating model performance —-

```r
yelp_test_pred <- predict(rating_classifier, yelp_test)
yelp_test_pred_prob <- predict(rating_classifier, yelp_test, type = "raw")

head(yelp_test_pred_prob)
```

```
##                 1         5
## [1,] 2.482516e-08 1.0000000
## [2,] 6.781768e-07 0.9999993
## [3,] 8.234301e-01 0.1765699
## [4,] 2.782953e-09 1.0000000
## [5,] 6.853143e-08 0.9999999
## [6,] 5.166816e-08 0.9999999
```

```r
library(gmodels)
CrossTable(yelp_test_pred, yelp_test_labels, prop.chisq = FALSE, prop.t = FALSE,
    prop.r = FALSE, dnn = c("predicted", "actual"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  1226
##
##
##              | actual
##    predicted |         1 |         5 | Row Total |
## -------------|-----------|-----------|-----------|
##            1 |       160 |        72 |       232 |
##              |     0.669 |     0.073 |           |
## -------------|-----------|-----------|-----------|
##            5 |        79 |       915 |       994 |
##              |     0.331 |     0.927 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       239 |       987 |      1226 |
##              |     0.195 |     0.805 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```r
## Accuracy
(160 + 915)/1226
```

```
## [1] 0.8768352
```

```r
## Sensitivity
915/987
```

```
## [1] 0.9270517
```

```r
## Specificity
160/239
```

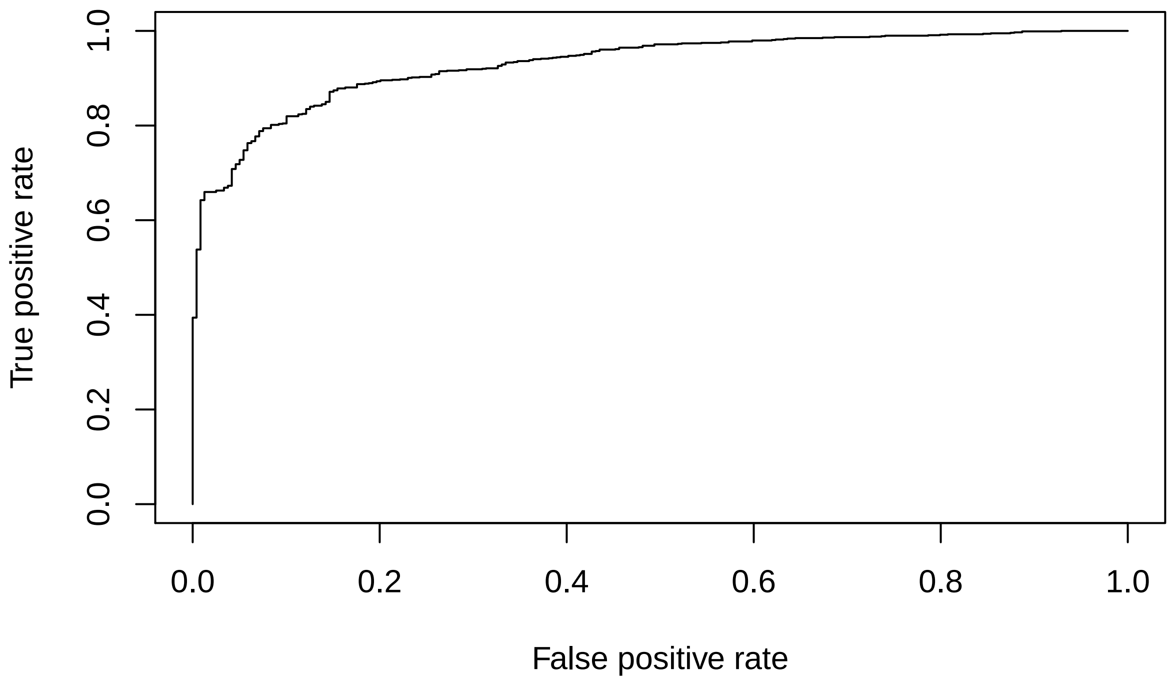
```
## [1] 0.6694561
```

We see that the sensitivity is approx. 0.93 and the sensitivity is approx. 0.67. Thus, the model has a much easier time detecting five-star reviews than one-star reviews.

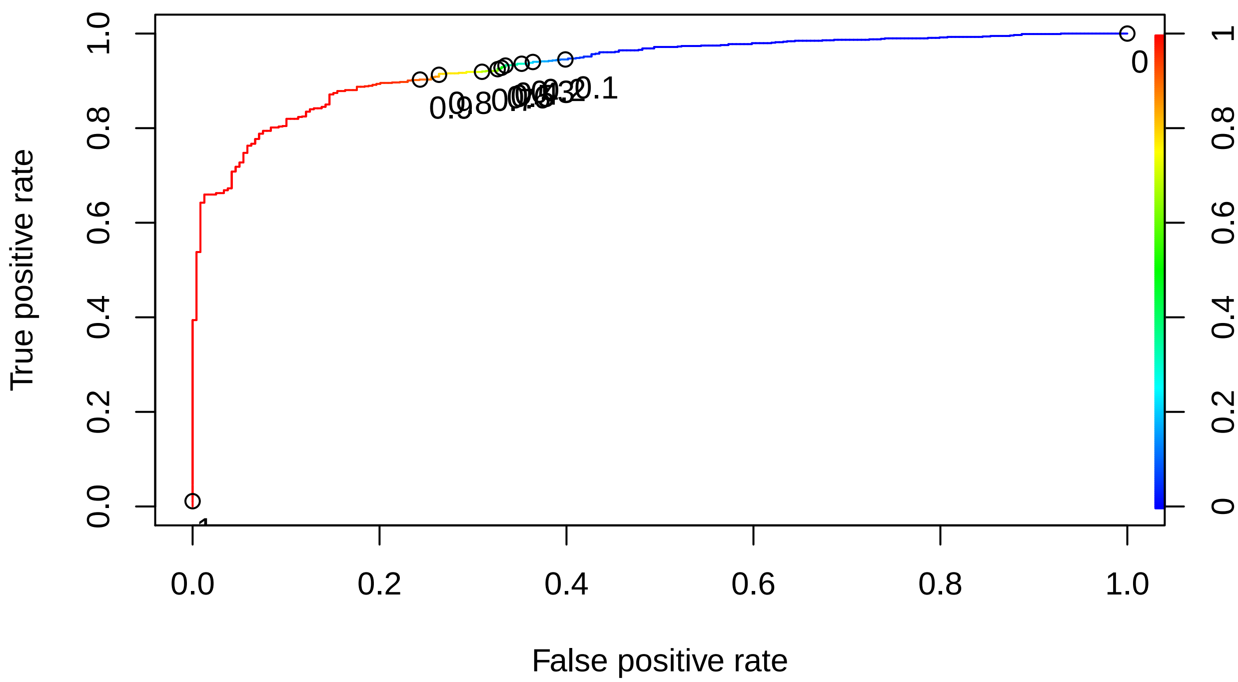
```r
## Plotting the ROC
library(ROCR)
yelp_test_pred_prob <- predict(rating_classifier, yelp_test, type = "raw")
ROCRpredTest = prediction(yelp_test_pred_prob[, 2], yelp_test_labels == "5")
ROCRperf <- performance(ROCRpredTest, "tpr", "fpr")
str(ROCRperf)
```

```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "False positive rate"
##   ..@ y.name      : chr "True positive rate"
##   ..@ alpha.name  : chr "Cutoff"
##   ..@ x.values    :List of 1
##   .. ..$ : num [1:1208] 0 0 0 0 0 0 0 0 0 0 ...
##   ..@ y.values    :List of 1
##   .. ..$ : num [1:1208] 0 0.0111 0.0152 0.0182 0.0203 ...
##   ..@ alpha.values:List of 1
##   .. ..$ : num [1:1208] Inf 1 1 1 1 ...
```

```r
plot(ROCRperf)
```




```r
plot(ROCRperf, colorize = TRUE)
plot(ROCRperf, colorize = TRUE, print.cutoffs.at = seq(0, 1, 0.1), text.adj = c(-0.2,
    1.7))
```




```r
## Area Under the curve (AUC)
perf.auc <- performance(ROCRpredTest, measure = "auc")
str(perf.auc)
```

```
## Formal class 'performance' [package "ROCR"] with 6 slots
##   ..@ x.name      : chr "None"
##   ..@ y.name      : chr "Area under the ROC curve"
##   ..@ alpha.name  : chr "none"
##   ..@ x.values    : list()
##   ..@ y.values    :List of 1
##   .. ..$ : num 0.931
##   ..@ alpha.values: list()
```

```
unlist(perf.auc@y.values)
```

```
## [1] 0.9309178
```

Although the model gives a high auc (0.931), the tpr is higher than the true negative rate. We can balance the sensitivity and the specificity by selecting the optimum threshold(cutoff) for predicting a 5-star review.

Step 5: Improving model performance

```
opt.cut = function(perf, pred) {
    cut.ind = mapply(FUN = function(x, y, p) {
        d = (x - 0)^2 + (y - 1)^2
        ind = which(d == min(d))
        c(sensitivity = y[[ind]], specificity = 1 - x[[ind]], cutoff = p[[ind]])
    }, perf@x.values, perf@y.values, pred@cutoffs)
}
print(opt.cut(ROCRperf, ROCRpredTest))
```

```
##                  [,1]
## sensitivity 0.8713273
## specificity 0.8535565
## cutoff      0.9880202
```

At a threshold of approximately 0.988, the sensitivity and specificity are both approximately 0.86. This classifier can be used to classify the reviews(just the text) as 1- or 5-star which can later be used predict the rating a user would assign to a business. Laterally, it could be used by businesses to improve and achieve higher ratings.
