Lab 4: Yelp Reviews

Kelly Xie

Apr 11, 2017

1. Summary Statistics

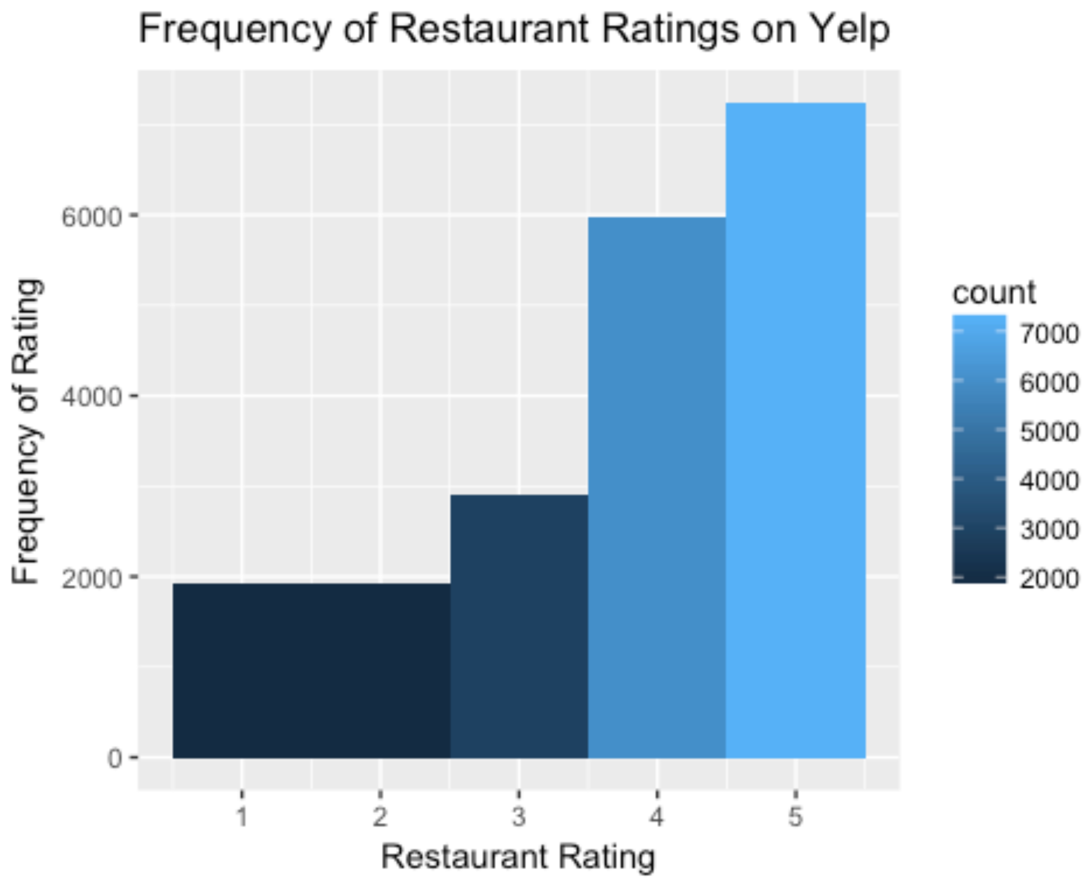
```r
# a histogram of all restaurant ratings given by users

#install.packages("ggplot2")
library("ggplot2")

reviews = read.csv("~/Desktop/YelpReviews_20k.csv")
reviews = unique(reviews) #remove duplicate reviews

gg = ggplot(data=reviews, aes(reviews$stars)) +
  geom_histogram(binwidth = 1, aes(fill = ..count..)) +
  xlab("Restaurant Rating") +
  ylab("Frequency of Rating") +
  ggtitle("Frequency of Restaurant Ratings on Yelp")

gg
```




```
# calculate the number of reviews the average restaurant in this sample recei
ved

#install.packages("dplyr")
library("dplyr")

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

byrestaurant = reviews %>%
  group_by(business_id) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
summarise(byrestaurant, "Average Number of Reviews by Restaurant" = mean(coun
t))
```

```
## # A tibble: 1 × 1
##   `Average Number of Reviews by Restaurant`
##                                       <dbl>
## 1                                  2.420441

# calculate the number of reviews the average user has contributed in this sa
mple

byuser = reviews %>%
  group_by(user_id) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
summarise(byuser, "Average Number of Reviews by User" = mean(count))

## # A tibble: 1 × 1
##   `Average Number of Reviews by User`
##                                 <dbl>
## 1                            1.152578

# on average, do GoodForLunch restaurants receive a greater number of reviews

goodforlunch = reviews %>%
  subset(GoodforLunch == "True") %>%
  group_by(business_id) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
summarise(goodforlunch, "Average Number of Reviews (GoodForLunch)" = mean(cou
nt))

## # A tibble: 1 × 1
##   `Average Number of Reviews (GoodForLunch)`
##                                        <dbl>
## 1                                   2.189089

notgood = reviews %>%
  subset(GoodforLunch == "False") %>%
  group_by(business_id) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
summarise(notgood, "Average Number of Reviews (NOT GoodForLunch)" = mean(coun
t))

## # A tibble: 1 × 1
##   `Average Number of Reviews (NOT GoodForLunch)`
##                                            <dbl>
## 1                                       2.590708
```

On average, restaurants marked Good For Lunch receive 0.4 (15.5%) fewer reviews than restaurants that are not.

```r
# on average, do GoodForLunch restaurants receive a higher number of stars

GFL = reviews %>%
  subset(GoodforLunch == "True") %>%
  group_by(business_id) %>%
  summarise(rating = mean(stars)) %>%
  arrange(desc(rating))
summarise(GFL, "Average Rating (GoodForLunch)" = mean(rating))
```

```
## # A tibble: 1 × 1
##   `Average Rating (GoodForLunch)`
##                             <dbl>
## 1                        3.617724
```

```r
notGFL = reviews %>%
  subset(GoodforLunch == "False") %>%
  group_by(business_id) %>%
  summarise(rating = mean(stars)) %>%
  arrange(desc(rating))
summarise(notGFL, "Average Rating (NOT GoodForLunch)" = mean(rating))
```

```
## # A tibble: 1 × 1
##   `Average Rating (NOT GoodForLunch)`
##                                 <dbl>
## 1                            3.605655
```

On average, restaurants marked Good For Lunch on Yelp are rated 0.012 (0.33%) higher than restaurants that are not.

2. Exploratory Text Analysis

```r
#convert reviews to text corpus
#install.packages("tm")
library("tm")
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
corp.original = VCorpus(VectorSource(reviews$text))

#clean and reprocess the text
corp = tm_map(corp.original, removePunctuation)
corp = tm_map(corp, removeNumbers)
corp = tm_map(corp, content_transformer(removeWords), stopwords("SMART"),lazy
```

```
=TRUE)
corp = tm_map(corp, content_transformer(tolower),lazy=TRUE)
corp = tm_map(corp, content_transformer(stemDocument),lazy=TRUE)
corp = tm_map(corp, stripWhitespace)

#generate a document-term matrix
dtm = DocumentTermMatrix(corp)
m = as.matrix(dtm)

#get fifteen most frequently appearing words among all reviews
word.freq = colSums(m)
word.freq = sort(word.freq, decreasing=TRUE)
as.data.frame(word.freq[1:15])

##         word.freq[1:15]
## food              15778
## good              13901
## place             13769
## order             10041
## great              9642
## veri               8774
## time               8479
## servic             8032
## tri                6432
## back               5876
## realli             5870
## restaur            5642
## friend             5311
## love               5199
## onli               4388

#generate a word cloud using the document-term matrix (max 100 words)
#size of the word correlates to its frequency in the review

#install.packages("wordcloud")
library("wordcloud")

## Loading required package: RColorBrewer

wordcloud(names(word.freq), word.freq, scale = c(4, .5),
          max.words = 100, colors = brewer.pal(6, "Dark2"), random.order = FA
LSE)
```

3. Text Analytics and Prediction

```r
#get number of unique terms in document-term matrix
dim(dtm)

## [1] 19988 34255
```

There are 34255 unique terms in the document-term matrix.

```r
#narrow the list down to the 200 words with the most predictive power

dtms = removeSparseTerms(dtm, .990) #remove sparse terms with .990 threshold
dtm_matrix = as.matrix(dtms)

#calculate correlation matrix between document-term matrix and goodforlunch
corr = cor(as.numeric(as.logical(reviews$GoodforLunch)), dtm_matrix)
absCorr = abs(corr) #get absolute value of correlations

#keep 200 terms with highest correlation magnitudes (both pos and neg)
top200 = order(absCorr, decreasing=TRUE)[1:200]
top200words = colnames(absCorr)[top200]
```

```r
#create new document-term matrix with these terms
newDTM.df = as.data.frame(cbind(GoodForLunch = as.numeric(as.logical(reviews$
GoodforLunch)),
                             dtm_matrix[,top200words]))
newDTM.m = as.matrix(newDTM.df)
dim(newDTM.m)

## [1] 19988   201
```

There are 201 unique terms in the new document-term matrix.

```r
#generate a wordcloud where the size corresponds to the correlation strength
#of the top 20 positive and negative words

#get top 20 positive words
top20pos = order(corr, decreasing=TRUE)[1:20]
top20poswords = colnames(corr)[top20pos]
pos.df = as.data.frame(cbind(term = top20poswords, corr = corr[top20pos]))

#get top 20 negative words
top20neg = order(corr)[1:20]
top20negwords = colnames(corr)[top20neg]
neg.df = as.data.frame(cbind(term = top20negwords, corr = corr[top20neg]))

#form wordcloud
wordcloud(words = c(as.character(pos.df$term), as.character(neg.df$term)),
          freq = c(as.numeric(as.character(pos.df$corr)), abs(as.numeric(as.c
haracter(neg.df$corr)))),
          scale = c(2.5, .5),
          colors = c(rep("green",20), rep("blue",20)),
          ordered.colors = TRUE, random.order = FALSE, random.color = FALSE)
```

Legend: Positive term = green; Negative term = blue

```
#Partition the matrix into training and test rows so you can use the test dat
a to evaluate your model performance. Set the last 20% of your rows aside for
testing, and use the first 80% to build your model as specified below
traindata = newDTM.df[1:(.8*nrow(newDTM.df)),]
testdata = newDTM.df[-(1:(.8*nrow(newDTM.df))),]

#Fit a logistic regression model to the selected variables in the training da
ta.
model = glm(GoodForLunch ~ ., data = traindata, family = binomial)
model

##
## Call:  glm(formula = GoodForLunch ~ ., family = binomial, data = traindata
)
##
## Coefficients:
## (Intercept)       reserv      dessert    breakfast     sandwich
##  -3.180e-01   -1.040e+00   -2.614e-01   -4.745e-01    3.041e-01
##        wine        night       servic        lunch       burger
##  -2.789e-01   -2.090e-01   -1.481e-01    2.298e-01    2.034e-01
##     chicken        steak        filet          fri       dinner
##   1.608e-01   -3.314e-01   -1.150e+00    1.441e-01   -9.434e-02
```

```
##       waiter       server        noodl        coffe          dog 
##   -1.638e-01   -9.503e-02    4.819e-01   -3.040e-01    4.054e-01 
##         view       brunch       chines       pancak    atmospher 
##   -1.034e+00   -4.643e-01    2.770e-01   -6.119e-01   -2.571e-01 
##         taco        drink       beauti       experi       chocol 
##    1.625e-01   -8.167e-02   -3.054e-01   -1.262e-01   -2.507e-01 
##         rice        veggi       butter     birthday       attent 
##    1.679e-01    3.729e-01   -3.812e-01   -1.118e-01   -1.260e-01 
##     cocktail          hot      ambianc      authent          pho 
##   -2.137e-01    1.010e-01   -4.490e-01    2.608e-01    3.045e-01 
##         fast       potato       celebr         tabl         vega 
##    2.753e-01   -1.723e-01   -6.848e-01    7.062e-02   -1.339e-01 
##        clean         cake        entre         chef          egg 
##    3.272e-01   -1.498e-01   -2.110e-01   -1.476e-01   -3.018e-01 
##        cours      mexican         dine       salmon        start 
##   -8.707e-02    1.727e-01    1.217e-02   -3.050e-01   -9.304e-02 
##          bbq        happi        glass         morn        great 
##    1.282e-01   -1.242e-01    3.953e-02   -4.182e-01   -3.540e-02 
##          bar         seat      restaur          bun        light 
##   -1.284e-02    2.573e-02   -5.525e-02    1.837e-01   -1.752e-01 
##        appet         date      scallop        today      perfect 
##    2.945e-02   -2.970e-01   -2.274e-01    3.045e-01    2.741e-02 
##        combo        cream         beef       french        bacon 
##    6.161e-02   -1.031e-01    9.777e-02   -2.069e-01   -3.221e-01 
##      bartend         amaz         crab         pull        excel 
##   -1.358e-01   -9.025e-02   -1.370e-01    2.012e-01   -1.254e-01 
##        group        crave         room        toast      burrito 
##   -2.680e-02    3.739e-01   -5.205e-02   -1.337e-01    2.351e-01 
##         show         bowl        parti          end          sat 
##   -3.007e-01    1.001e-01   -8.010e-03   -1.881e-02   -9.502e-02 
##        bottl      counter         meat        place         bean 
##   -1.417e-01    1.624e-01    1.134e-01    8.718e-02    3.758e-02 
##     calamari        prime        music         mash         dish 
##   -3.855e-01   -2.237e-01   -2.186e-01   -1.721e-01   -9.010e-02 
##         pork         thai         veri         hour         main 
##   -2.512e-02    1.487e-01   -2.286e-02    1.491e-02    3.460e-02 
##        salsa        sushi       turkey        diner         menu 
##    9.205e-02   -1.084e-01    2.421e-01   -2.216e-01    4.576e-03 
##      present      fantast         hous         nice       finish 
##   -2.610e-02   -1.588e-01   -6.437e-02   -2.461e-02   -1.462e-01 
##         soda         sauc        onion        waffl      deliveri 
##    2.572e-01    9.441e-02    8.538e-02   -1.930e-01    2.112e-01 
##     charlott        share         meal       lettuc     tortilla 
##    5.867e-01   -7.131e-02    1.188e-02   -1.448e-02    2.785e-01 
##         felt         tuna        decor         wait        quick 
##   -9.655e-02   -1.269e-01    1.199e-02   -4.023e-03    2.092e-01 
##        arriv      seafood        bread          guy        pickl 
##    2.179e-01   -1.957e-01   -7.388e-02    1.684e-01    9.847e-02 
##      healthi          set          app        hotel          cut 
##    3.402e-01   -2.904e-02   -4.910e-01   -3.778e-02   -1.827e-01 
```

```
##        patio      impress         list     knowledg        fruit
##   -7.831e-02   -1.430e-01   -4.464e-02    4.713e-02    5.609e-03
##        curri         lamb       expens         ring        worth
##   -3.642e-02   -3.121e-01   -2.834e-01    1.076e-01   -1.515e-01
##          add         cool        strip         dark         late
##    1.464e-01   -2.404e-01   -5.616e-02   -1.591e-01   -2.213e-01
##       oyster        joint        pasta        short         pour
##   -8.602e-02    1.448e-01   -2.594e-02    1.307e-02   -1.976e-01
##      hostess          box      weekend        split        fresh
##    3.579e-02    3.393e-01   -4.483e-02   -2.149e-01    1.692e-01
##      lobster      comfort        relax       casino        locat
##    1.504e-01   -1.660e-01   -2.586e-01   -3.807e-02    4.347e-02
##          run        guest         vibe          leg         oliv
##    2.171e-01   -9.527e-02   -2.371e-01    4.525e-02   -1.924e-01
##       banana     saturday      spinach        water         soup
##    1.253e-01   -5.698e-03   -1.179e-01   -3.324e-02    1.153e-02
##     outstand        enjoy       friday        befor     waitress
##    1.183e-02    8.100e-02    3.038e-02    1.911e-03   -5.483e-02
##          fun       desert      outdoor       greasi         fine
##   -1.239e-01   -1.917e-01   -2.245e-01    1.974e-02   -1.141e-01
##        floor        plate      surpris         rich         book
##   -2.802e-01   -2.806e-02   -1.066e-01    1.531e-01   -1.647e-01
##       overal      delight       valley         talk          eat
##   -9.203e-05   -1.764e-01    5.799e-01   -7.310e-02    1.021e-01
##         trip
##   -1.549e-01
##
## Degrees of Freedom: 15989 Total (i.e. Null);  15789 Residual
## Null Deviance:       21300
## Residual Deviance: 18650     AIC: 19050
```

```
#A positive coefficient positively predicts that a restaurant is good for lun
ch.
#A negative coefficient suggests a restaurant would not be good for lunch.

#Use the coef command to access top positive and negative words from the mode
l.
coef = coef(model)[-1]
pos.terms = coef[coef>0]
top.pos = sort(pos.terms,decreasing=T)[1:15]
top.pos
```

```
##  charlott    valley     noodl       dog     crave    veggi   healthi
## 0.5866726 0.5798723 0.4819046 0.4053690 0.3738928 0.3728794 0.3401948
##       box     clean     today       pho  sandwich  tortilla    chines
## 0.3393160 0.3271950 0.3045467 0.3045441 0.3041261 0.2784771 0.2770426
##      fast
## 0.2752966
```

```r
neg.terms = coef[coef<0]
top.neg = sort(neg.terms)[1:15]
top.neg
```

```
##      filet     reserv       view     celebr     pancak        app 
## -1.1499466 -1.0404650 -1.0341077 -0.6848086 -0.6118874 -0.4909538 
##  breakfast     brunch    ambianc       morn   calamari     butter 
## -0.4745300 -0.4643414 -0.4489867 -0.4181817 -0.3855215 -0.3812082 
##      steak      bacon       lamb 
## -0.3313668 -0.3221268 -0.3120732
```

```r
#Produce a word cloud that separates the top 15 positive words and top 15 neg
ative words.

poswords = tibble::rownames_to_column(as.data.frame(top.pos), var="term")
negwords = tibble::rownames_to_column(as.data.frame(top.neg), var="term")

#form wordcloud
wordcloud(words = c(poswords$term, negwords$term),
          freq = c(poswords$top.pos, abs(negwords$top.neg)),
          scale = c(4.5, .5),
          colors = c(rep("red",15), rep("purple",15)),
          ordered.colors = TRUE, random.order = FALSE, random.color = FALSE)
```

Legend: Positive words = red; Negative words = purple

```
#Using the model you have generated, choose a probability threshold to maximi
ze accuracy and classify the restaurants in your training data as 1 or 0 acco
rding to whether they are GoodForLunch.
traindata$predict_val = predict(model, type="response")
traindata$gfl_predicted = traindata$predict_val > 0.5

#How well does this model perform on the training data in terms of classifica
tion accuracy (i.e. the percentage of GoodForLunch values that you get correc
t)?
accuracy = sum(traindata$gfl_predicted == traindata$GoodForLunch) / nrow(trai
ndata)
accuracy

## [1] 0.6843027
```

With a probability threshold of 0.5, this model has a 68.43% classification accuracy for predicting GoodForLunch in the training data.

```
#Predict values for GoodforLunch in your test data.
testdata$predict_val = predict(model, newdata = testdata, type = "response")
testdata$gfl_predicted = testdata$predict_val > 0.5

#How well does the model perform in terms of classification accuracy (i.e. th
e percentage of GoodForLunch values that you get correct)?
accuracy.test = sum(testdata$gfl_predicted == testdata$GoodForLunch) / nrow(t
estdata)
accuracy.test

## [1] 0.6783392
```

With a probability threshold of 0.5, this model has a 67.83% classification accuracy for predicting GoodForLunch in the test data.