Analyzing text in Yelp reviews - Text mining, Sentiment Analysis

Abhinav Ram Bhatta, Prajwal Prashanth, Anviksha Gupta

2022-11-20

```
library(dplyr)
library(ggplot2)
library(tidytext)
library(stringr)
library(gridExtra)
library(textstem)
library(textdata)
library(tidyr)
library(caret)
library(SnowballC)
library(tidyverse)
library(e1071)
library(ranger)
library(rsample)
library(pROC)
library(magrittr)
library(RColorBrewer)
library(viridis)
```

Data Exploration:

Question 1

Explore the data. (a) How does star ratings for reviews relate to the star-rating given in the dataset for businesses (attribute 'businessStars')? Can one be calculated from the other? (b) Here, we will focus on star ratings for reviews. How are star ratings distributed? How will you use the star ratings to obtain a label indicating 'positive' or 'negative' – explain using the data, summaries, graphs, etc.?

```
df = read.csv2("/Users/abhinavram/Documents/IDS572 Data Mining/Assignment 3/yelpResta
urantReviews_sample_f22.csv", sep =';')
glimpse(df)
```

Star ratings will be used here to indicate the sentiment label. For binary classification, we will convert the 1-5 scale rating values to : positive (1) & negative (0)

Distribution of star ratings

```
head(df)
```

```
##                  review_id                    user_id             business_id
```

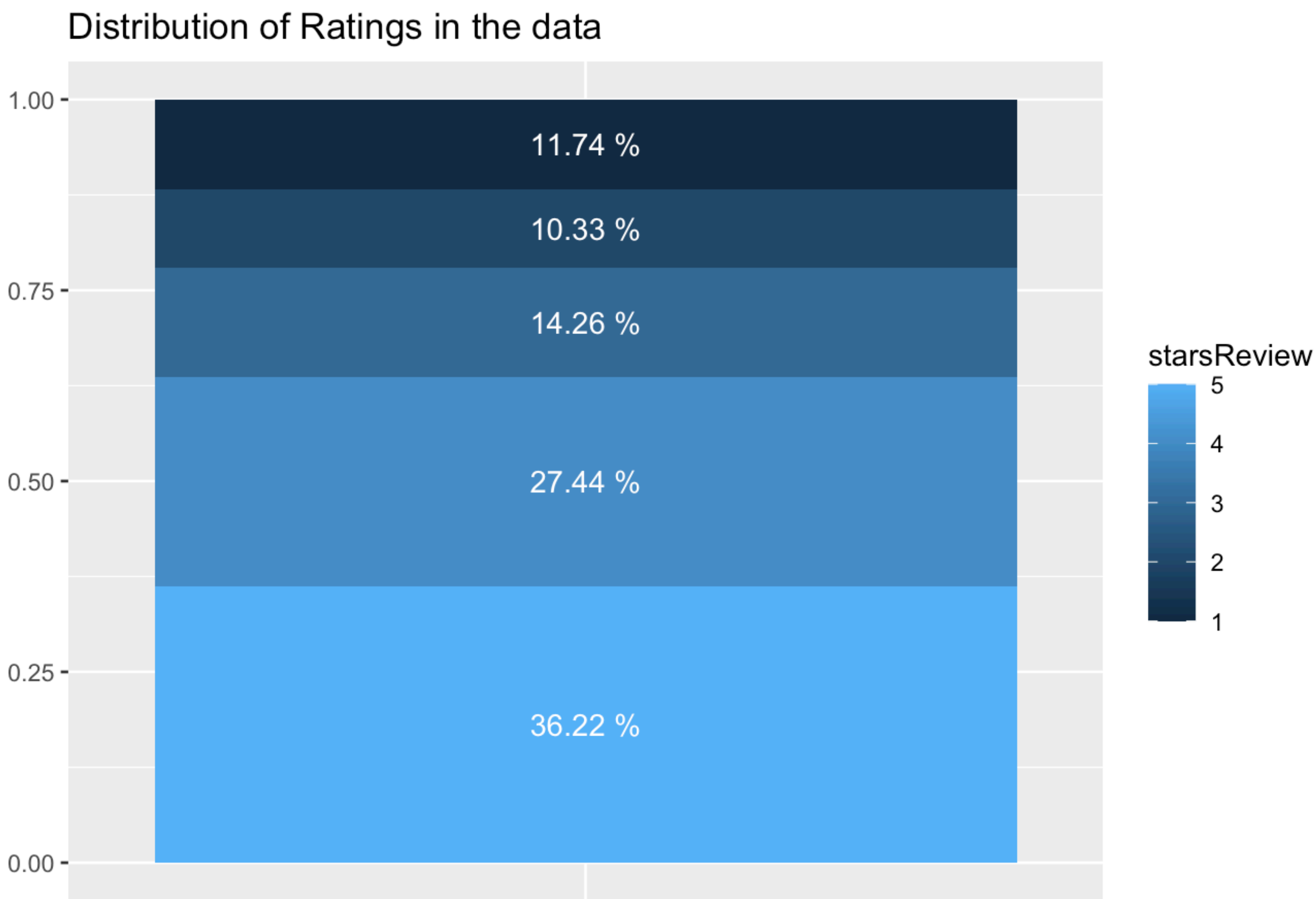
```
## 1 K24CBfrL8nQXlGOmFInmVw hyIVFPfm3TyPWQf0Xh9u1Q PSUOncuqfqHulYj_fusthw
## 2 245Q0ZiJESuxuPzJNHuuoQ gEfBCDtfQC9-HGT76l_orQ h-Oq86DZfZad9kKXe8m7Lg
## 3 SAPHmaFcO-nIVUU-1HB-ig OVni5hOjD-wP4nxtxNn4AA T1A2zbyPfwGrJgG_8hAirQ
## 4 hwhIX1JdifD_PZCKtxuKUw QY4o6hOYgBcgmLpbcdsbBw T1A2zbyPfwGrJgG_8hAirQ
## 5 6dlamZjo03C6CvSwCB3S-g tDb-EDd6MV26tGwk6U8QOw 7jl9EJQCGBiTyXTDXMsRIw
## 6 nEQawGIHIbzPuHa7jSc2Yg vxRgc6S1mgXbZGXOE5nylQ 4xnVH2jTVwrsO88q_jHWWw
##   starsReview       date
## 1           4 2010-07-24
## 2           4 2015-05-13
## 3           2 2011-10-25
## 4           2 2016-03-13
## 5           5 2015-01-02
## 6           5 2016-01-26
##
text
## 1
Cheeburger is my go to burger shop, it's not too far from my house, and the food is p
retty solid.\n\nWhere else can you get a burger with Eleventy Billion toppings, prett
y solid O-rings, and a malt, served by friendly staff in a pretty timely manner?\n\nT
he Best way to sum up Cheeburger is imagine if a Fuddruckers and a Malt Shop had a ba
by, that my friends is Cheeburger.\n\nMy only major complaint is that for a burger, o
-rings, and a malt, the tab can be a little high, but the quality makes up for it.\n\
nThere are probably a million different combos of toppings/sauces you can put on your
burger, and they will custom blend any flavor of malt that you can dream up. They als
o have Eggcreams (my NY roommate said they were pretty damn good), and a few other de
licacies that you can't get at many places in town.\n\nIf you're on the SW side of to
wn and looking for a good burger fix, it's hard to do better than Cheeburger, and mad
props if you do the 1lb burger!!!
## 2
I've driven by Reyes on southern for a long time since that location opened Always cu
rious and so glad that we finally checked it out The torta del Rey for ten bucks is r
idiculously big and so tasty And their aguas frescas are so fresh and good Authentic
for sure
## 3
I'm typically a Marriott girl, so staying at a Hilton was a stretch! I must say, even
though the price was better here, I would still pay more for a Marriott property. The
front desk receptionist didn't even know how to get back to the Villas, where we were
staying. The Villa itself is nice, but that's about where it ends. Food at the restau
rant is good; there is better elsewhere in Scottsdale. It is conveniently located tho
ugh; and you can walk to The Good Egg in the strip mall adjacent. (Also - try Humble
Pie!). The friendliest people there were the concierge (great golf recommendations) a
nd the night person at the desk, who gladly split our bill between two couples. I wou
ldn't stay here again, unless it was another great deal.
## 4
I was here for a conference so sadly I didn't have a choice for another selection due
to the expenses being paid for by my employer.\nOverall it seems as if this Hilton is
struggling with a remodel (as of Feb 2016). It's as if they were limited in budget so
there were some upgrades, like the wallpaper and possibly the headboards but not the
desk or the dresser. \n\nWhen I first arrived at my room and opened my door I was gre
eted to the thick smell of chlorine... and my room was located no where near the pool
```

! Although they upgraded the TV the digital TV/cable system was something from the ea
rly 2000's. Most modern systems allow things like in-room checkout and services from
the TV but although it was advertised while watching the programs whenever you clicke
d on the remote to view it it took you to a login prompt which was clearly reserved f
or the IT admins. ???? \nI opened the cabinet to where the fridge should be my reacti
on was much like the scene from Shawshank Redemption where the warden discovers an es
cape tunnel hidden by a poster in Tim Robbin's cell. Where was the fridge? When I cal
led the front desk they mentioned how they were removing the mini-bars from the rooms
. If I wanted a fridge it was an extra $75 but was soon waived within 2 min after cal
ling me back.\n\nThe conference rooms were actually not bad. I only ate once at the r
estaurant on site which wasn't bad. There are several good food joints across the str
eet or at the mall one parking lot just to the south. If you need cheap snacks the sm
all strip mall across Scottsdale Ave has a Trader Joes tucked in back.\n\nI definitel
y wouldn't ever pay money to stay at this "resort" and I'm glad I didn't have to espe
cially after seeing all of the other choices (e.g. Doubletree, etc) nearby.\n\nThe on
e saving grace which kept it from being a single star is the service.
5
Came here to celebrate my cousin's birthday. The place has a dark décor with very sop
histicated waiters dressed well. Since you are at a top notch steak house, you will b
e dropping some serious dough here. All the steaks are prime and delicious. I was ver
y surprised that a steak house did not have any steak tartar or carpaccio. I was very
much in the mood for it that night and felt a little let down. The butternut squash s
oup was very tasty and full of delicious heavy cream. They also had some amazing oyst
ers, my wife commenting, better than the ones we had in Seattle and Portland. I had t
he Kansas City Bone in New York strip and ate the entire thing. I was actually pretty
amazed how much I piled in that night. I would love to be able to come here at least
once a week but I don't think my wallet will be able to afford it. Parking is complim
entary valet, you just need to tip at the end.
6 Hands down one the best Pizza establishments in the Cleveland Heights area!!! \n
\nPositives:\n\n* Ton of pizza choices (gourmet or custom) and unlimited toppings inc
luded in the price when you make your own\n* Prices are very reasonable for the quali
ty and quantity you get\n* Ingredients are top notch: from veggies to meat, everythin
g feels and tastes fresh \n* I have tried only the regular crust and that is yum, it
holds the toppings and still feels soft when you take a bite\n* Friendly staff who gi
ve you suggestions (that are legit and not just so that you place your order)\n* You
can order online (yay when I don't want to speak to a human being)\n\nNegatives:\n\n*
Delivery can take time during weekends so plan accordingly\n* If you look at their we
bsite/Facebook, they have a lot of deals, however the Cleveland Heights location does
not participate in a few of them (3 pizzas for $20)\n* Pizza prices are for two pies
and you can subtract $5 if you want one (they want you to order more). This is a litt
le confusing on the website\n* Their pizza sizes are smaller than the regular places
you are used to (Pizza Hut, Dominos). A Large is actually a Medium, and an X Large is
a Large so order a size up\n\nI have done a take out once and delivery a bunch of tim
es. Each and every time my pizza has been delectable to the point I are stuffed but w
ant to eat another slice. I have tried the Bombay Pizza, French Quarter gourmet pizza
s and the rest have been custom made. I have to say the chicken in the Bombay Pizza i
s an almost great tandoori chicken - it has the red color, slightly spicy, slightly t
angy. The sauce adds to the flavor and the banana peppers are just about the right ad
dition: everything works together. The French Quarter pizza tastes like something I h
ave eaten in NOLA. The cajun seasoning along with the andouille sausage and pepper/on

```
ions tastes like a part of a jumbalaya. As for the custom pizzas, most of them have h
ave been made with the spicy garlic sauce and a ton of veggies. The great thing is th
at even if you order all the items on the list, they know how much to put to make the
pizza tasty rather than only edible. Kudos to the people assembling the stuff in the
kitchen. The spicy garlic sauce is definitely one of my favorite sauces there, which
is why I keep getting it. \n\nOverall, this is a high quality pizza place, that is op
en late (especially on weekends), delivers in the Cleveland Heights area and satisfie
s all your cravings with a simple slice of pizza. I have yet to try the Miracle crust
and the Chicago style gourmet pizza. That is next on my list.... Definitely a 5 star
place as in my opinion the positives far outweigh the negatives.
##   useful funny cool                             name neighborhood
## 1      2     2    2            Cheeburger Cheeburger    Southwest
## 2      1     0    0             Los Reyes De La Torta         <NA>
## 3      1     0    1 Hilton Scottsdale Resort & Villas         <NA>
## 4      1     0    0 Hilton Scottsdale Resort & Villas         <NA>
## 5      0     0    1       Donovan's Steak & Chop House         <NA>
## 6      3     1    1                         pizzaBOGO         <NA>
##                address       city state postal_code latitude  longitude
## 1  8390 S Rainbow Blvd  Las Vegas    NV       89139 36.03598 -115.24300
## 2  1528 E Southern Ave      Tempe    AZ       85282 33.39301 -111.91400
## 3 6333 N Scottsdale Rd Scottsdale    AZ       85250 33.53073 -111.92454
## 4 6333 N Scottsdale Rd Scottsdale    AZ       85250 33.53073 -111.92454
## 5  3101 E Camelback Rd    Phoenix    AZ       85016 33.50947 -112.01552
## 6      13434 Cedar Rd  Cleveland    OH       44118 41.50111  -81.55686
##   starsBusiness review_count is_open
## 1           3.5          256       0
## 2           4.0          164       1
## 3           3.0          123       1
## 4           3.0          123       1
## 5           4.0          273       1
## 6           4.0           56       1
##
attributes
## 1                           Alcohol: none|Ambience: {'romantic': False, 'intimate
': False, 'classy': False, 'hipster': False, 'divey': False, 'touristy': False, 'tren
dy': False, 'upscale': False, 'casual': True}|BikeParking: True|BusinessAcceptsCredit
Cards: True|BusinessParking: {'garage': False, 'street': False, 'validated': False, '
lot': True, 'valet': False}|Caters: False|DriveThru: False|GoodForKids: True|HasTV: T
rue|NoiseLevel: average|OutdoorSeating: False|RestaurantsAttire: casual|RestaurantsDe
livery: False|RestaurantsGoodForGroups: True|RestaurantsPriceRange2: 2|RestaurantsRes
ervations: False|RestaurantsTableService: True|RestaurantsTakeOut: True|WheelchairAcc
essible: True|WiFi: no|GoodForMeal: {'dessert': False, 'latenight': False, 'lunch': T
rue, 'dinner': True, 'breakfast': False, 'brunch': False}
## 2                                                                Alcohol: beer_and
_wine|Ambience: {'romantic': False, 'intimate': False, 'classy': False, 'hipster': Fa
lse, 'divey': False, 'touristy': False, 'trendy': False, 'upscale': False, 'casual':
True}|BikeParking: True|BusinessAcceptsCreditCards: True|BusinessParking: {'garage':
False, 'street': False, 'validated': False, 'lot': True, 'valet': False}|Caters: True
|GoodForKids: True|GoodForMeal: {'dessert': False, 'latenight': False, 'lunch': True,
'dinner': True, 'breakfast': False, 'brunch': False}|HasTV: True|NoiseLevel: average|
```

```
OutdoorSeating: False|RestaurantsAttire: casual|RestaurantsDelivery: False|Restaurant
sGoodForGroups: True|RestaurantsPriceRange2: 2|RestaurantsReservations: False|Restaur
antsTableService: True|RestaurantsTakeOut: True|WiFi: no
## 3
<NA>
## 4
<NA>
## 5 Alcohol: full_bar|Ambience: {'romantic': False, 'intimate': False, 'classy': Tru
e, 'hipster': False, 'divey': False, 'touristy': False, 'trendy': False, 'upscale': T
rue, 'casual': False}|BikeParking: True|BusinessAcceptsCreditCards: True|BusinessPark
ing: {'garage': False, 'street': False, 'validated': False, 'lot': False, 'valet': Tr
ue}|Caters: False|GoodForKids: False|GoodForMeal: {'dessert': False, 'latenight': Fal
se, 'lunch': False, 'dinner': True, 'breakfast': False, 'brunch': False}|HasTV: False
|NoiseLevel: average|OutdoorSeating: False|RestaurantsAttire: dressy|RestaurantsDeliv
ery: False|RestaurantsGoodForGroups: True|RestaurantsPriceRange2: 4|RestaurantsReserv
ations: True|RestaurantsTableService: True|RestaurantsTakeOut: False|WheelchairAccess
ible: True|WiFi: no|GoodForDancing: False|HappyHour: True
## 6                                                Alcohol: none|Ambience: {'romantic':
False, 'intimate': False, 'classy': False, 'hipster': False, 'divey': False, 'tourist
y': False, 'trendy': False, 'upscale': False, 'casual': True}|BikeParking: True|Busin
essAcceptsCreditCards: True|BusinessParking: {'garage': False, 'street': True, 'valid
ated': False, 'lot': True, 'valet': False}|Caters: True|GoodForKids: True|GoodForMeal
: {'dessert': False, 'latenight': False, 'lunch': False, 'dinner': True, 'breakfast':
False, 'brunch': False}|HasTV: False|NoiseLevel: average|OutdoorSeating: False|Restau
rantsAttire: casual|RestaurantsDelivery: True|RestaurantsGoodForGroups: True|Restaura
ntsPriceRange2: 2|RestaurantsReservations: False|RestaurantsTableService: False|Resta
urantsTakeOut: True|WheelchairAccessible: True|WiFi: no
##                                                                categories
## 1                                                       Burgers|Restaurants
## 2                                                       Restaurants|Mexican
## 3 Restaurants|Hotels|Hotels & Travel|Event Planning & Services|Resorts
## 4 Restaurants|Hotels|Hotels & Travel|Event Planning & Services|Resorts
## 5                            Steakhouses|Seafood|Restaurants|Bars|Nightlife
## 6                                                         Pizza|Restaurants
##
hours
## 1     Monday 11:0-21:0|Tuesday 11:0-21:0|Wednesday 11:0-21:0|Thursday 11:0-21:0|Fr
iday 11:0-21:0|Saturday 11:0-21:0|Sunday 11:0-21:0
## 2  Monday 10:0-20:0|Tuesday 10:0-20:30|Wednesday 10:0-20:30|Thursday 10:0-20:30|Fr
iday 10:0-21:30|Saturday 9:0-22:30|Sunday 9:0-20:0
## 3                     Monday 0:0-0:0|Tuesday 0:0-0:0|Wednesday 0:0-0:0|Thursday 0:0-
0:0|Friday 0:0-0:0|Saturday 0:0-0:0|Sunday 0:0-0:0
## 4                     Monday 0:0-0:0|Tuesday 0:0-0:0|Wednesday 0:0-0:0|Thursday 0:0-
0:0|Friday 0:0-0:0|Saturday 0:0-0:0|Sunday 0:0-0:0
## 5                        Monday 16:0-22:0|Tuesday 16:0-22:0|Wednesday 16:0-22:0|Thur
sday 16:0-22:0|Friday 16:0-22:0|Saturday 16:0-22:0
## 6 Monday 10:30-23:0|Tuesday 10:30-23:0|Wednesday 10:30-23:0|Thursday 10:30-23:0|Fr
iday 10:30-0:0|Saturday 10:30-0:0|Sunday 12:0-22:0
```

```
#How are star ratings distributed?
tbl = df %>% group_by(starsReview) %>% count() %>% ungroup() %>% mutate(per=`n`/sum(`
n`)) %>% arrange(desc(starsReview))
tbl$label = paste(round(tbl$per*100,2),"%")
dist_chart=ggplot(data=tbl)+geom_bar(aes(x="", y=per, fill=starsReview), stat="identi
ty", width = 1)+ geom_text(aes(x=1, y = cumsum(per) - per/2, label=label),color="whit
e") + xlab("")+ylab("")+ggtitle("Distribution of Ratings in the data")
dist_chart
```

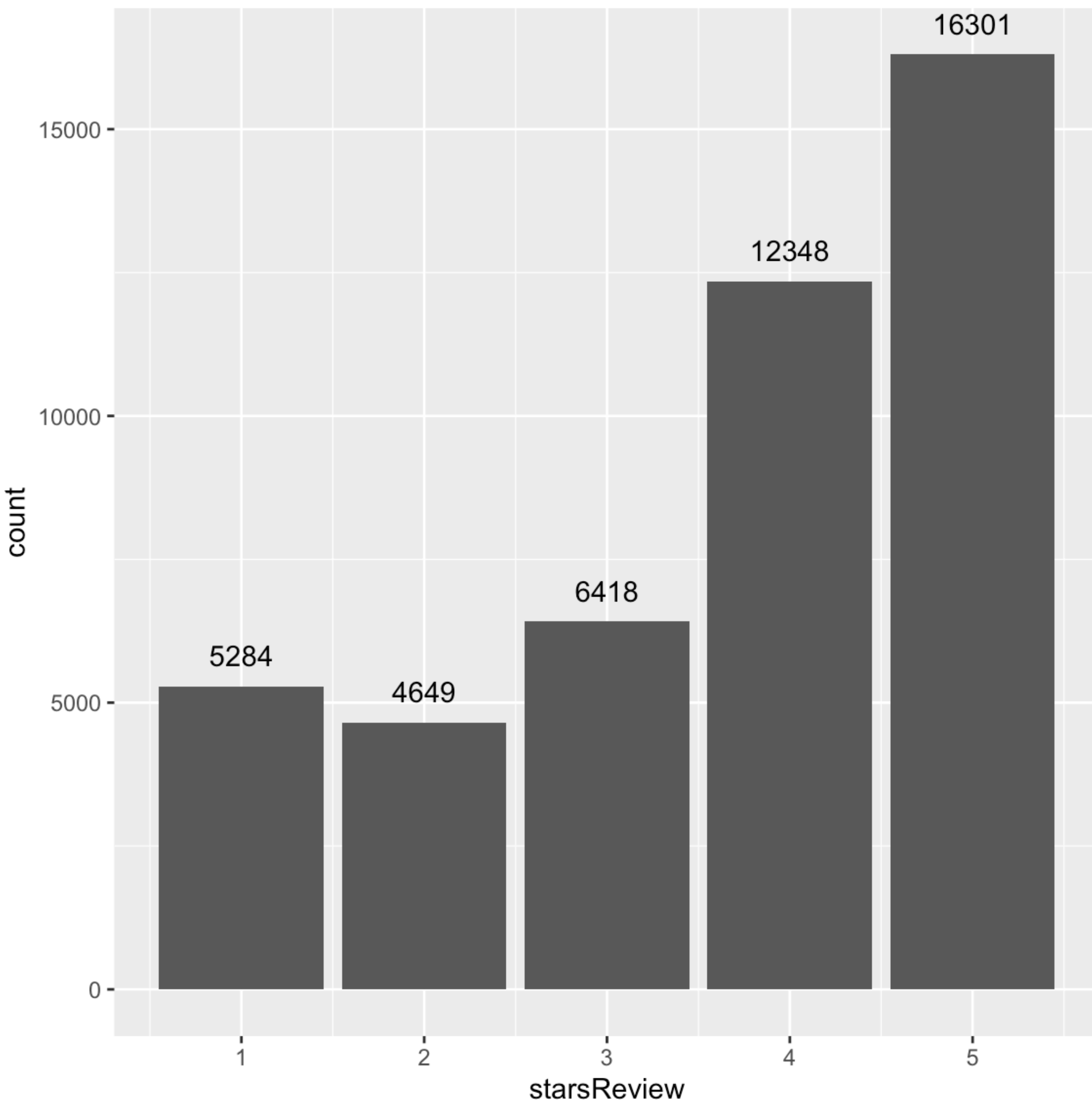



```
df %>% group_by(state) %>% tally()
```

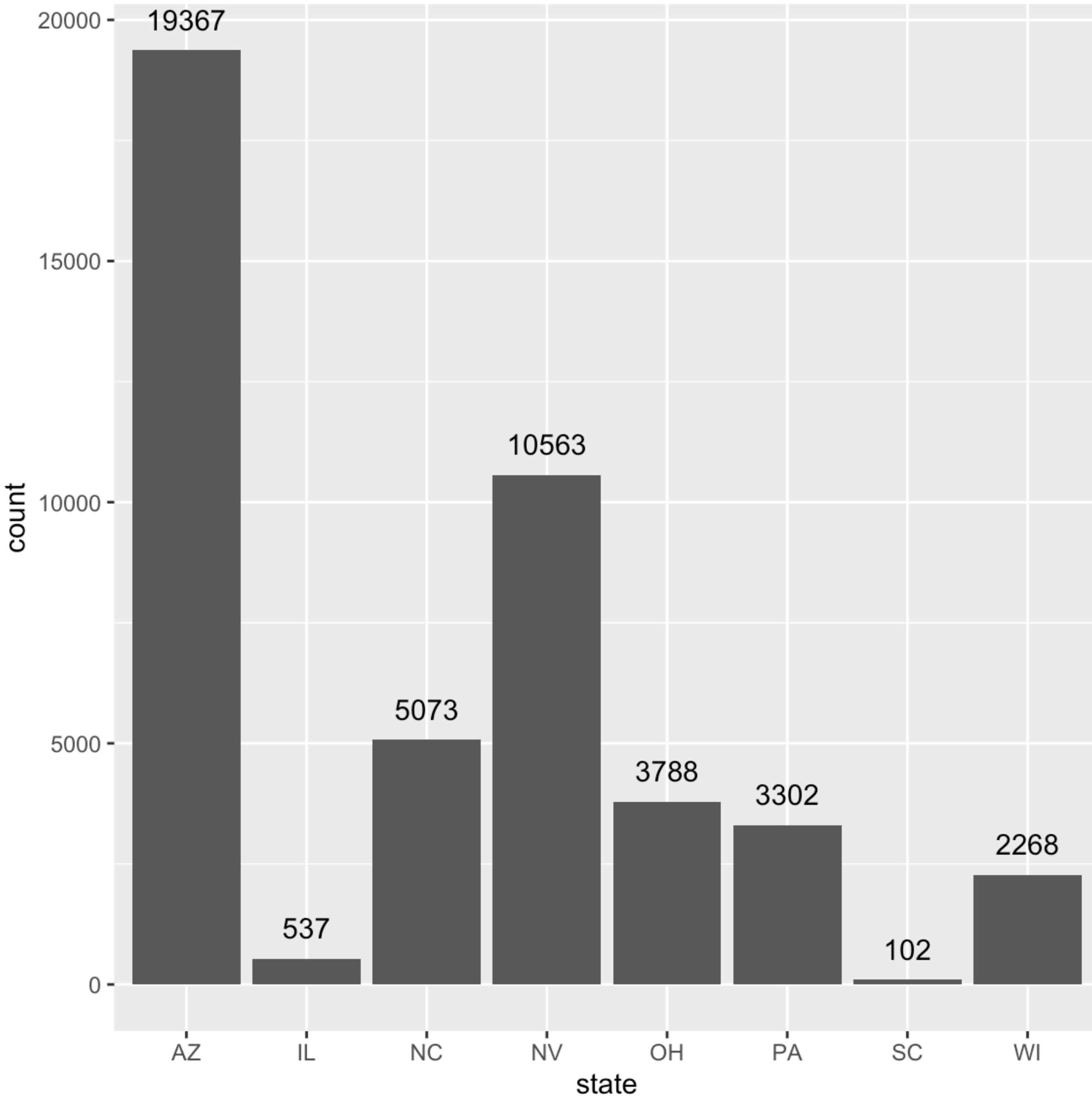
```
## # A tibble: 8 × 2
##   state     n
##   <chr> <int>
## 1 AZ    19367
## 2 IL      537
## 3 NC     5073
## 4 NV    10563
## 5 OH     3788
## 6 PA     3302
## 7 SC      102
## 8 WI     2268
```

From the histogram, we can see that the number of reviews is increasing from star rating 1 to 5. Star ratings 4 and 5 account for most of the total reviews - 64.56%, while the star ratings 1,2, and 3 account for remaining with almost the same proportion each.

```
ggplot(df, aes(x = starsReview)) + geom_bar() + geom_text(stat='count', aes(label=..c
ount..), vjust=-1)
```




```
#Most of the reviews are 5 star, almost 38%
ggplot(df, aes(x=state)) + geom_bar() + geom_text(stat='count', aes(label=..count..),
vjust=-1)
```



We observe that most of the reviews are from Arizona, Nevada and the least reviews are from South Carolina, Illinois with less than 500 reviews. Below histogram shows distribution of reviews from different states

```
#1,2 : Negative
#4,5 : Positive
rrDF = df %>% filter(str_detect(postal_code, "^[0-9]{1,5}"))
#tokenize the text of the reviews in the column named 'text'
df_Tokens = rrDF %>% unnest_tokens(word, text)
df_Tokens %>% distinct(word) %>% dim()
#remove stopwords
df_Tokens = df_Tokens %>% anti_join(stop_words)
df_Tokens %>% distinct(word) %>% dim()
#lets remove the words which are not present in at least 10 reviews
rareWords = df_Tokens %>% count(word, sort=TRUE) %>% filter(n<10)
rareWords %>% distinct(word) %>% dim()
df_Tokens = anti_join(df_Tokens, rareWords) %>% filter(str_detect(word,"[0-9]")==FALS
E)
#Term-frequency, tf-idf
df_Tokens = df_Tokens %>%  mutate(word = textstem::lemmatize_words(word))
df_Tokens = df_Tokens %>% filter(str_length(word)>=3 & str_length(word)<=15)
df_Tokens = df_Tokens %>% group_by(review_id, starsReview) %>% count(word)
df_Tokens = df_Tokens %>% bind_tf_idf(word, review_id, n)
```

Question 2

What are some words in the restaurant reviews indicative of positive and negative sentiment – identify at least 20 in each category. One approach for this is to determine the average star rating for a word based on star ratings of documents or reviews where the word occurs. Do these ‘positive’ and ‘negative’ words make sense in the context of user reviews for restaurants being considered? (For this, since we’d like to get a general sense of positive/negative terms, you may like to consider a pruned set of terms – say, those which occur in a certain minimum and maximum number of documents).

The usage of words in different star ratings are different to convey their message. We observe that words like “love”,“delicious”, “amazing”, “nice”,“friendly”,“pretty” ….etc are most used in 5,4-star ratings which show positive emotion.

Top words in 5-star reviews

```
#Which words are related to higher/lower star raings in general
df_Tokens %>% filter(starsReview==5) %>% filter(!word %in% c('food', 'time', 'restaur
ant', 'service')) %>% group_by(word) %>% count(word, sort=TRUE)
```

```
## # A tibble: 7,013 × 2
## # Groups:   word [7,013]
##    word          n
##    <chr>     <int>
##  1 love       4245
##  2 delicious  3704
##  3 friendly   3097
##  4 amaze      3038
##  5 eat        2806
##  6 fresh      2383
##  7 staff      2367
##  8 nice       2334
##  9 price      2246
## 10 menu       2205
## # … with 7,003 more rows
```

Top words in 4-star reviews

```
df_Tokens%>% filter(starsReview==4) %>% filter(!word %in% c('food', 'time', 'restaura
nt', 'service')) %>% group_by(word) %>% count(word, sort=TRUE)
```

```
## # A tibble: 7,078 × 2
## # Groups:   word [7,078]
##    word          n
##    <chr>     <int>
##  1 love       2627
##  2 nice       2450
##  3 eat        2391
##  4 price      2338
##  5 delicious  2328
##  6 menu       2218
##  7 friendly   2081
##  8 chicken    1935
##  9 pretty     1863
## 10 fry        1843
## # … with 7,068 more rows
```

Top words in 3-star reviews

```
df_Tokens%>% filter(starsReview==3) %>% filter(!word %in% c('food', 'time', 'restaura
nt', 'service')) %>% group_by(word) %>% count(word, sort=TRUE)
```

```
## # A tibble: 6,782 × 2
## # Groups:   word [6,782]
##    word        n
##    <chr>   <int>
##  1 eat      1396
##  2 nice     1345
##  3 pretty   1324
##  4 price    1282
##  5 menu     1192
##  6 taste    1103
##  7 bite     1076
##  8 chicken  1034
##  9 fry      1017
## 10 drink     965
## # … with 6,772 more rows
```

Top words in 2-star reviews

```
df_Tokens%>% filter(starsReview==2) %>% filter(!word %in% c('food', 'time', 'restaura
nt', 'service')) %>% group_by(word) %>% count(word, sort=TRUE)
```

```
## # A tibble: 6,371 × 2
## # Groups:   word [6,371]
##    word       n
##    <chr>  <int>
##  1 eat     1077
##  2 taste   1010
##  3 bad      970
##  4 wait     822
##  5 price    820
##  6 table    809
##  7 drink    759
##  8 menu     744
##  9 minute   719
## 10 nice     715
## # … with 6,361 more rows
```

Top words in 1-star reviews

```
df_Tokens%>% filter(starsReview==1) %>% filter(!word %in% c('food', 'time', 'restaura
nt', 'service')) %>% group_by(word) %>% count(word, sort=TRUE)
```

```
## # A tibble: 6,063 × 2
## # Groups:   word [6,063]
##    word       n
##    <chr>  <int>
##  1 bad     1535
##  2 eat     1386
##  3 wait    1131
##  4 minute  1022
##  5 table    901
##  6 taste    897
##  7 leave    882
##  8 tell     844
##  9 drink    826
## 10 people   764
## # … with 6,053 more rows
```

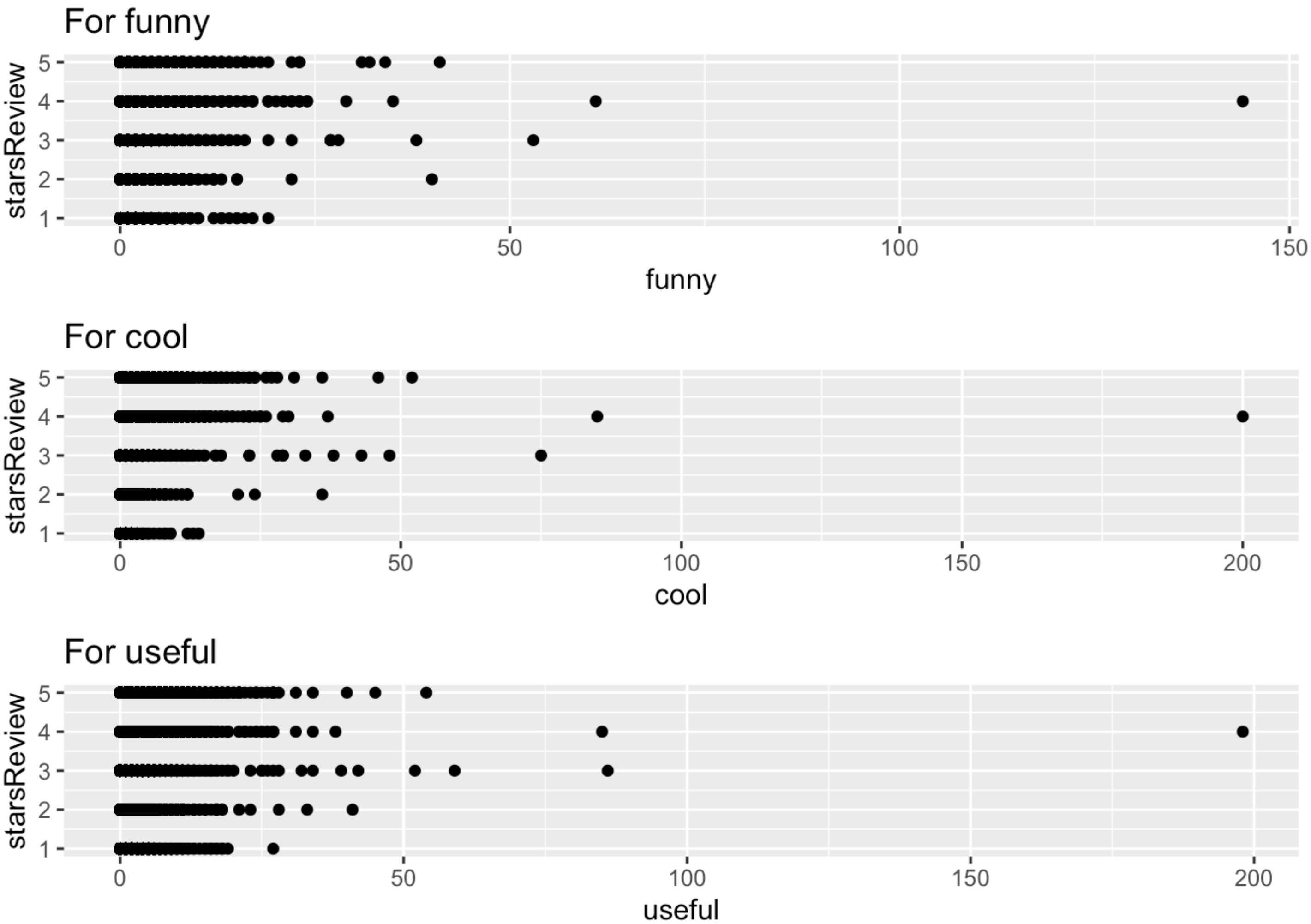
Here is the final data frame with lemmetized words, term-frequency and inverse term-frequencies calculated

```
head(df_Tokens)
```

```
## # A tibble: 6 × 7
## # Groups:   review_id, starsReview [1]
##   review_id              starsReview word          n    tf   idf tf_idf
##   <chr>                        <int> <chr>     <int> <dbl> <dbl>  <dbl>
## 1 __0PpIOWdiB5VG5NiHvQtQ           5 average       2  0.08  3.60  0.288
## 2 __0PpIOWdiB5VG5NiHvQtQ           5 brunch        1  0.04  4.06  0.162
## 3 __0PpIOWdiB5VG5NiHvQtQ           5 chai          1  0.04  6.44  0.258
## 4 __0PpIOWdiB5VG5NiHvQtQ           5 diner         1  0.04  4.17  0.167
## 5 __0PpIOWdiB5VG5NiHvQtQ           5 enjoyable     1  0.04  4.97  0.199
## 6 __0PpIOWdiB5VG5NiHvQtQ           5 family        1  0.04  2.90  0.116
```

Yelp users can vote a review as either funny, useful, or cool. Lets see the distribution of ratings across different vote categories. Below plot shows number of votes vs star ratings for each voting category.

```
plots_review = list()
plots_review[[1]] = ggplot(df, aes(x=funny , y=starsReview)) +geom_point() + ggtitle(
paste("For funny"))
plots_review[[2]] = ggplot(df, aes(x=cool , y=starsReview)) +geom_point() + ggtitle(p
aste("For cool"))
plots_review[[3]] = ggplot(df, aes(x=useful , y=starsReview)) +geom_point() + ggtitle
(paste("For useful"))
do.call(grid.arrange,plots_review)
```

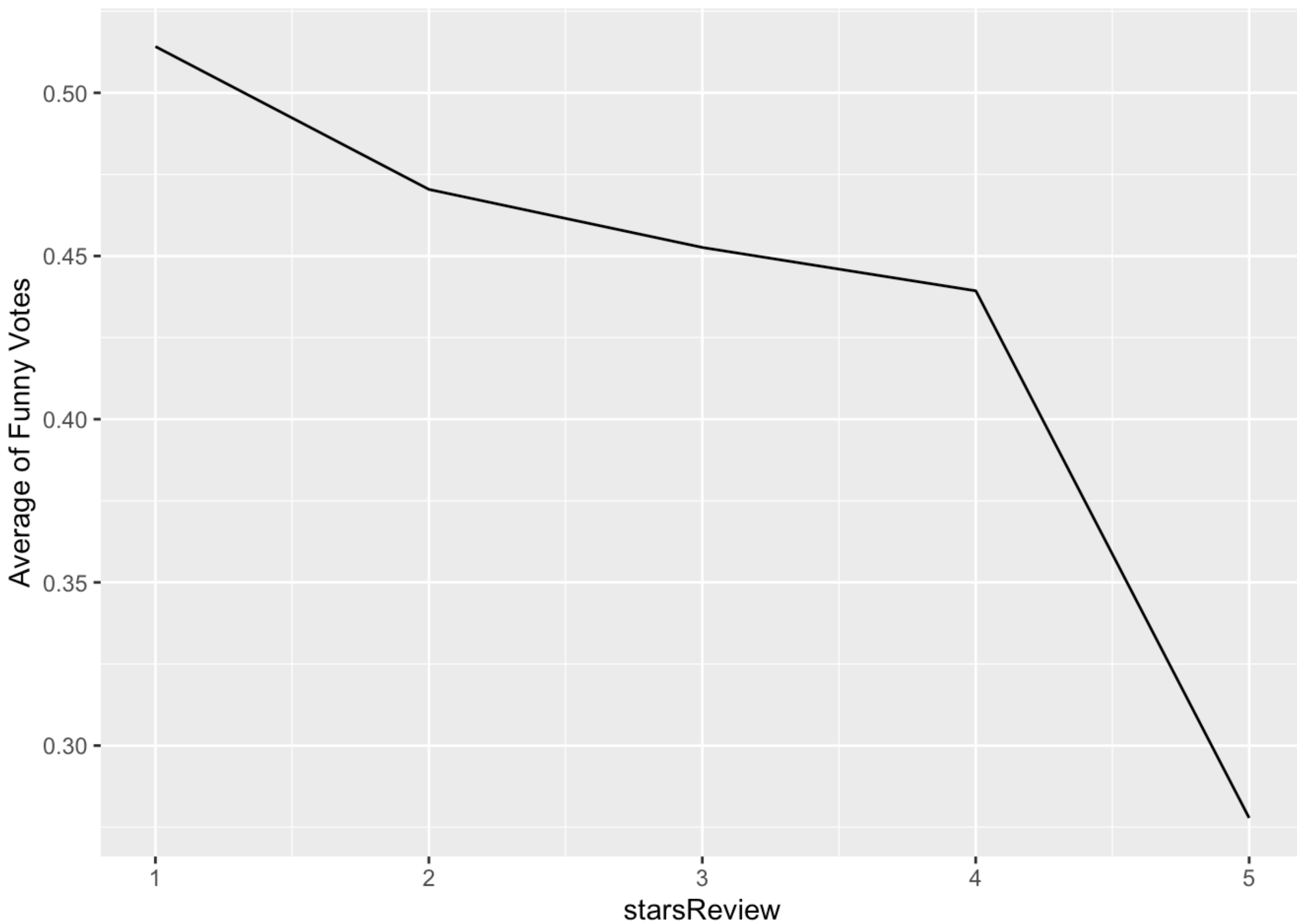


From the above plot, we can see a few reviews from star rating 3,4 have high votes for funny, cool, useful reviews. Now let’s see the average votes for each across star ratings.

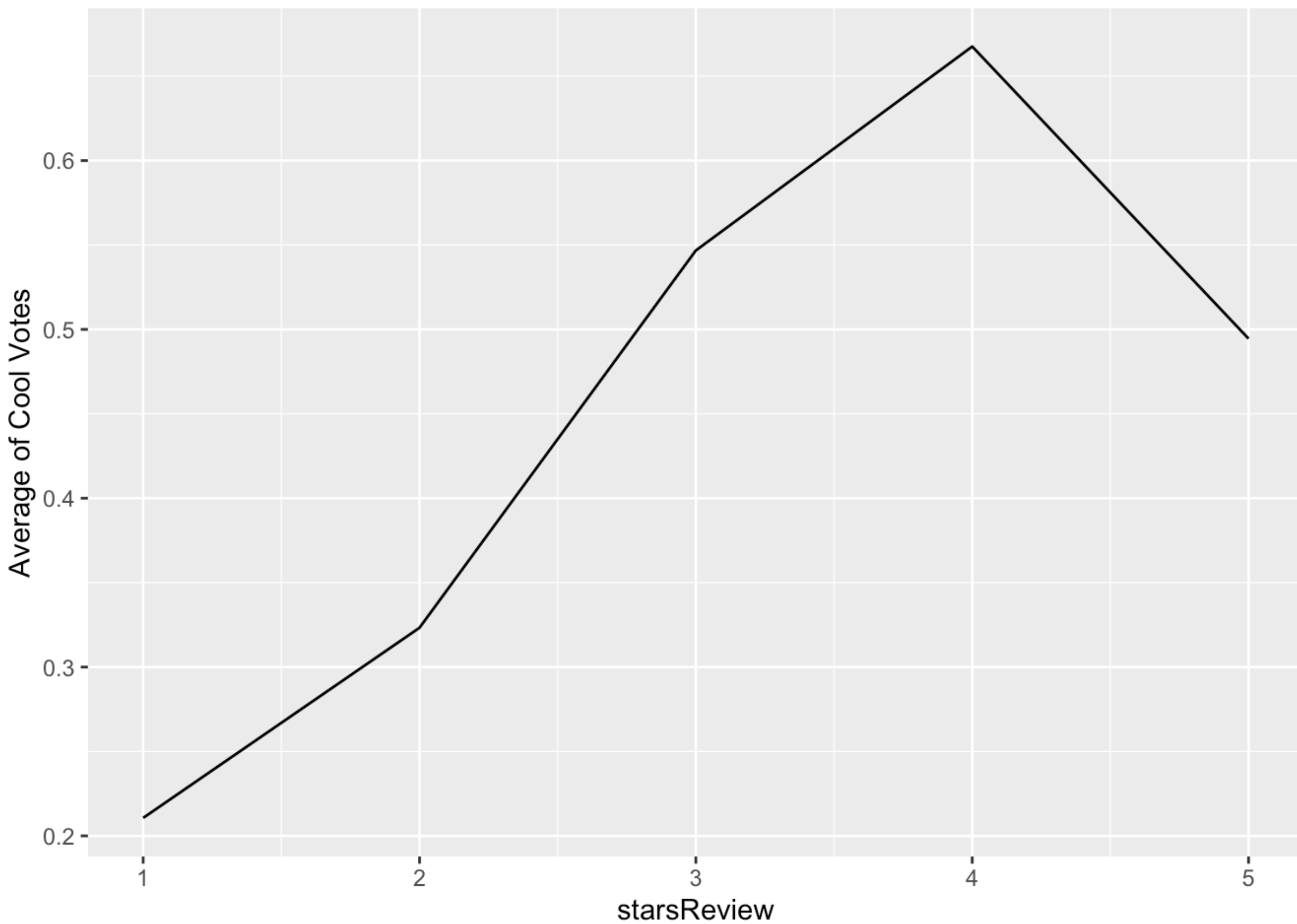
```
plots_review = list()
plots_review[[1]] = ggplot(df, aes(x= useful, y=funny)) +geom_point()
plots_review[[2]] = ggplot(df, aes(x= useful, y=cool)) +geom_point()
plots_review[[3]] = ggplot(df, aes(x= cool, y=funny)) +geom_point()
do.call(grid.arrange,plots_review)
```

The first line graph shows the average funny votes across star ratings and can be seen that low star ratings have funny comments. This could be because users tend to give sarcastic reviews when they hate the restaurant. So, funny reviews are associated with negative sentiment.

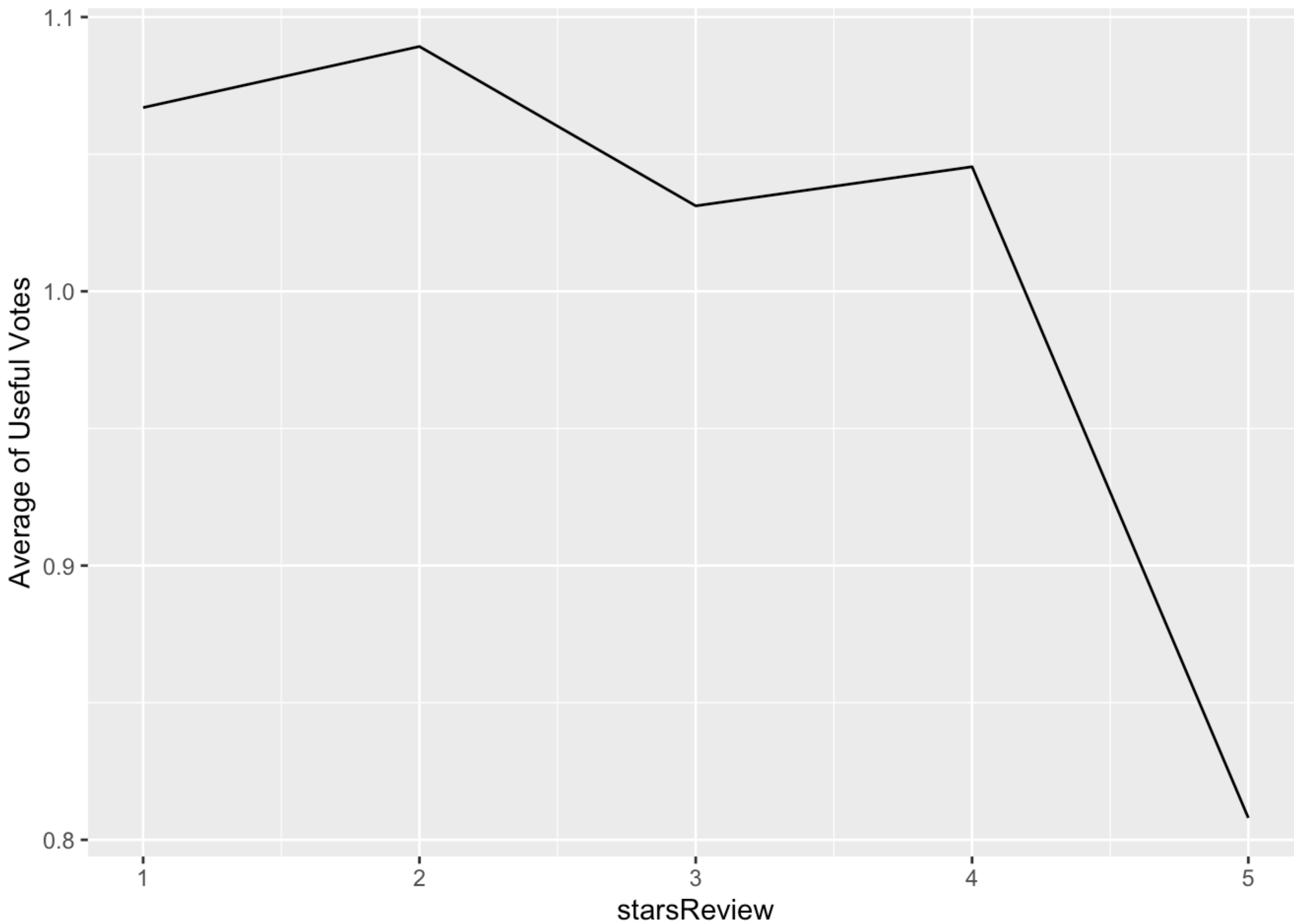
```
ggplot(df %>% group_by(starsReview) %>% summarize(AvgFunny_votes = mean(funny))) + ae
s(x=starsReview, y=AvgFunny_votes, fill=starsReview) + geom_line() + xlab("starsRevie
w") + ylab("Average of Funny Votes")
```




```
ggplot(df %>% group_by(starsReview) %>% summarize(AvgCool_votes = mean(cool))) + aes(
x=starsReview, y=AvgCool_votes, fill=starsReview) + geom_line() + xlab("starsReview")
+ ylab("Average of Cool Votes")
```




```
ggplot(df %>% group_by(starsReview) %>% summarize(AvgUseful_votes = mean(useful))) +
aes(x=starsReview, y=AvgUseful_votes, fill=starsReview) + geom_line() + xlab("starsRe
view") + ylab("Average of Useful Votes")
```

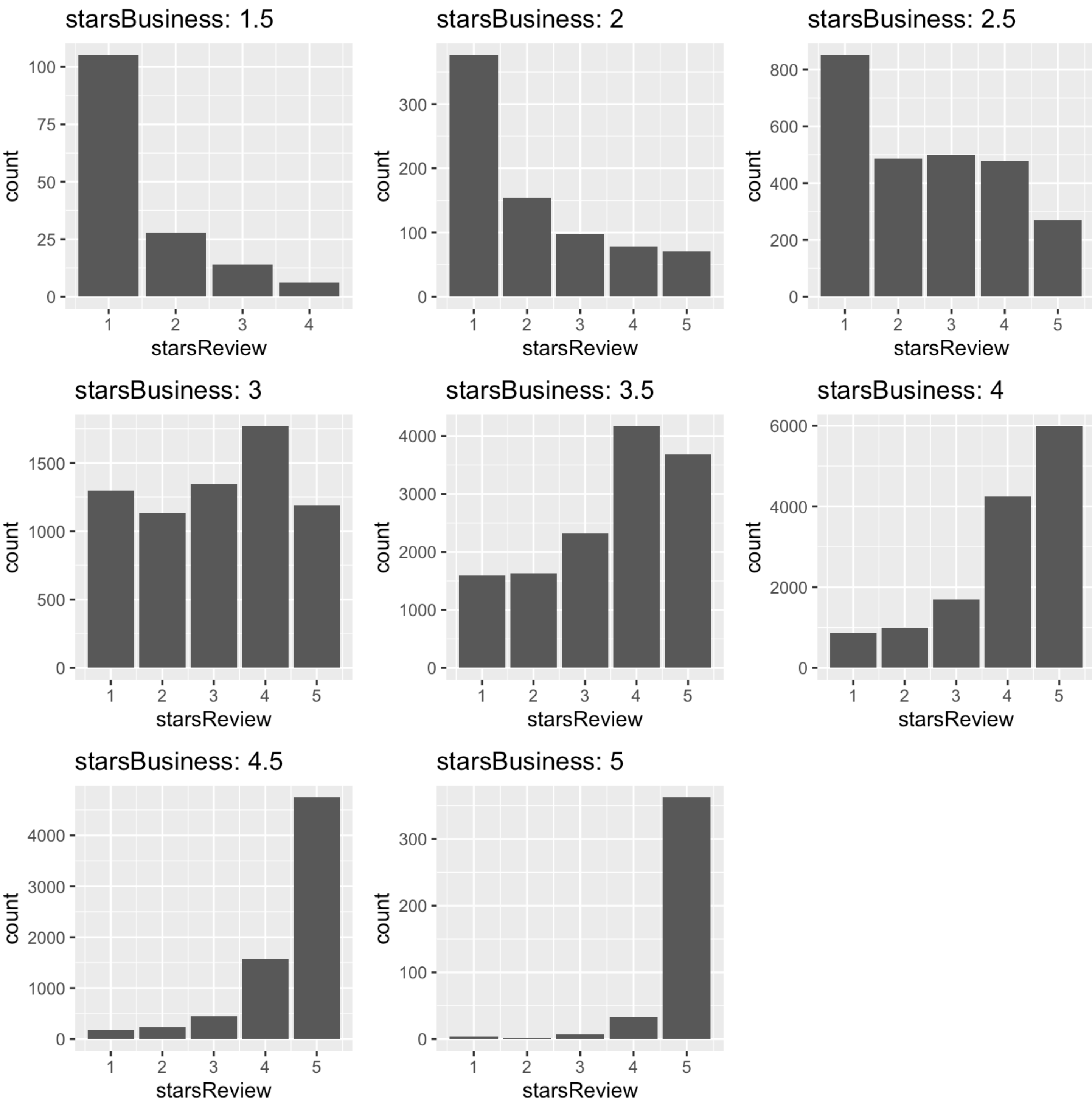


The second line graph shows the average cool votes across star ratings and can be seen that as the number of stars increases the cool comments increase but again decreases with a 5-star rating.

The third line graph shows the average useful votes across star ratings and low star rating reviews are voted useful when compared to high star ratings. This is expected because most of the high star ratings will have common reviews like – Food is good, served on time, and reviews about maintenance and ambiance which most of the users don't find useful.

Now lets see how does star ratings for reviews relate to the star-rating given in the dataset for business

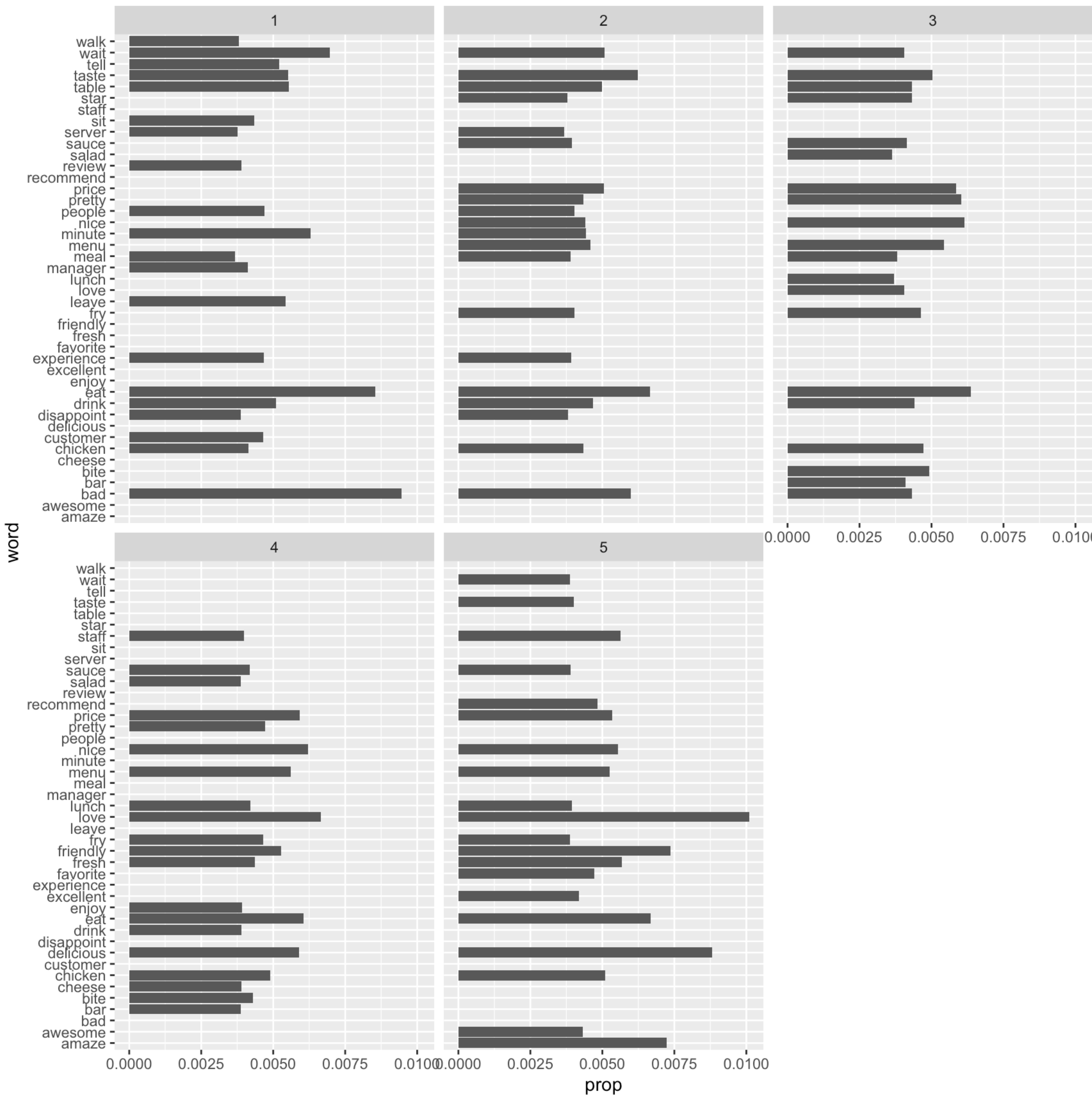
```
#How does star ratings for reviews relate to the star-rating given in the dataset for
business (attribute ‘businessStars’)?
p = list()
i = 1
for (r in c(1.5,2,2.5,3,3.5,4,4.5,5)){
  tbl_ = df[df$starsBusiness==r,]
  p[[i]] = ggplot(tbl_, aes(x = starsReview)) + geom_bar() + ggtitle(paste("starsBusi
ness:",r))
  i = i+1
}
do.call(grid.arrange,p)
```



In the above plot we can see that for starsBusiness rating of 1.5 most of the reviews are of 1-star and as the starsBusiness ratings increase the positive reviews keep increasing. Restaurants with starsBusiness rating > 4 are doing good with most of the ratings being 4,5-star.

Words including and not limited to “food”, “service”, “time” and “restaurant” are common in all the reviews so lets remove them and after eliminating them we have obtained the below graph which depicts the proportion of top wprds in each review.

```
tbl2 = df_Tokens%>% group_by(starsReview) %>% count(word, sort=TRUE) %>% mutate(prop=n/sum(n))
xx = tbl2 %>% group_by(word) %>% summarise(totWS=sum(starsReview*prop))
xx %>% top_n(20)
xx %>% top_n(-100)
tbl2 %>% filter(!word %in% c('food', 'time', 'restaurant', 'service')) %>% group_by(starsReview) %>% arrange(starsReview, desc(prop))%>% filter(row_number()<=20) %>% ggplot(aes(word, prop))+geom_col()+coord_flip()+facet_wrap((~starsReview))
```



In the above figure we can see that for 5 star rating the proportion of words like awesome, amazing, love, delicious and pretty is high. For 1 star rating the proportion for words like bad and wait is very high.

The below table shows the number of occurrences of a word in reviews 5 and 1.

Question 3

We will consider three dictionaries, available through the tidytext package – (i) the extended sentiment lexicon developed by Prof Bing Liu, (ii) the NRC dictionary of terms denoting different sentiments, and (iii) the AFINN dictionary which includes words commonly used in user-generated content in the web. The first specifies lists of positive and negative words, the second provides lists of words denoting different sentiment (for eg., positive, negative, joy, fear, anticipation, …), while the third gives a list of words with each word being associated with a positivity score from -5 to +5.

a. How many matching terms (i.e. terms in your data which match the dictionary terms) are there for each of the dictionaries?
b. What is the overlap in matching terms between the different dictionaries? Based on this, do you think any of the three dictionaries will be better at picking up sentiment information from you text of reviews?
c. Consider the positive and negative terms you determined in Q 2 above; which of these terms match with terms in each of the three dictionaries?

Lexicon Dictionaries

We will consider three dictionaries, available through the tidytext package The NRC dictionary of terms denoting different sentiments, The extended sentiment lexicon developed by Prof Bing Liu, and The AFINN dictionary which includes words commonly used in user-generated content in the web.

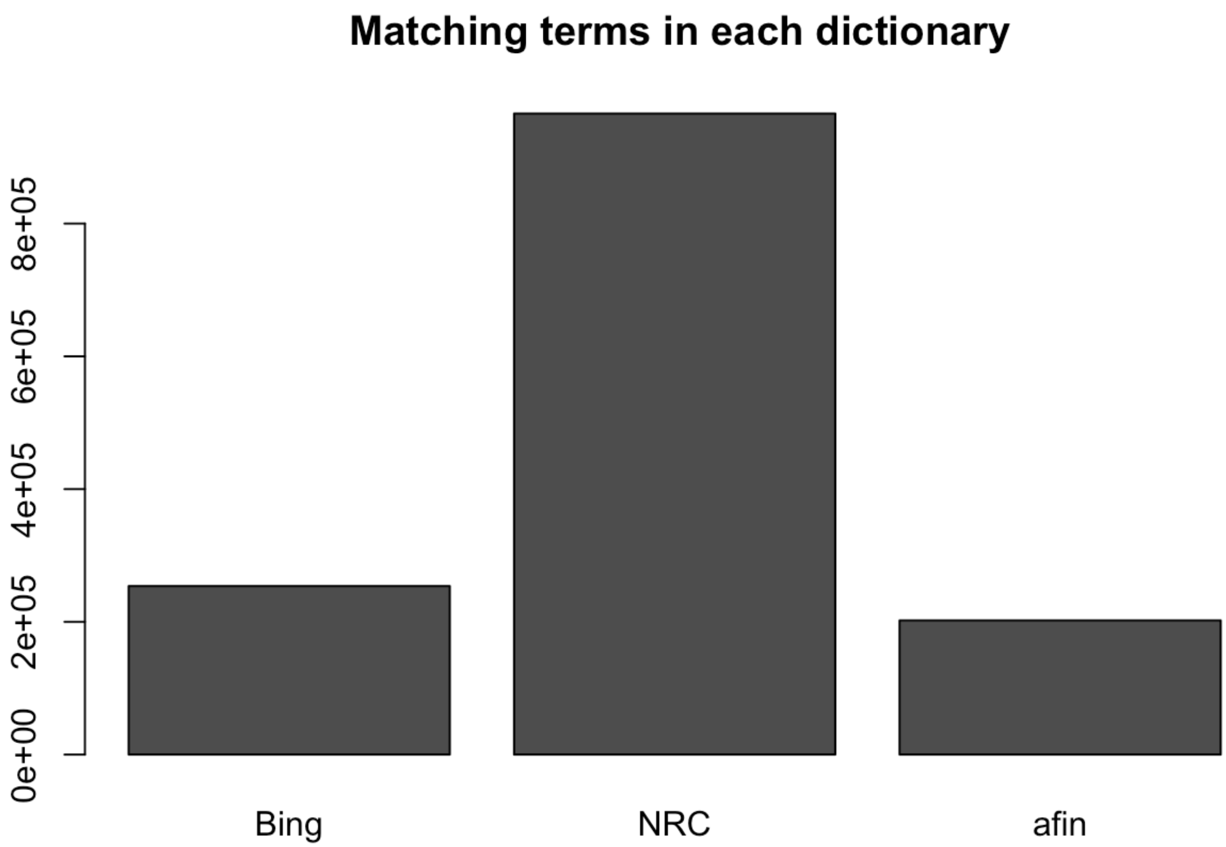
The first provides lists of words denoting different sentiment, the second specifies lists of positive and negative words, while the third gives a list of words with each word being associated with a positivity score from -5 to +5.

The number of matching terms for each dictionary is calculated and depicted using the graph below.

```
#How many matching terms are there for each of the dictionaries?
from_bing_dict = inner_join(get_sentiments("bing"),df_Tokens, by="word")
from_nrc_dict = inner_join(get_sentiments("nrc"),df_Tokens, by="word")
from_afin_dict = inner_join(get_sentiments("afinn"),df_Tokens, by="word")
binn = nrow(from_bing_dict)
nrcn = nrow(from_nrc_dict)
affn = nrow(from_afin_dict)
table_terms = matrix(c(binn,nrcn,affn),ncol=3,byrow=TRUE)
colnames(table_terms) = c("Bing","NRC","afin")
rownames(table_terms) = c("Number of matching terms")
table_terms = as.table(table_terms)
table_terms
```

```
##                            Bing    NRC   afin
## Number of matching terms 253963 965834 202167
```

```
barplot(table_terms, main =" Matching terms in each dictionary")
```

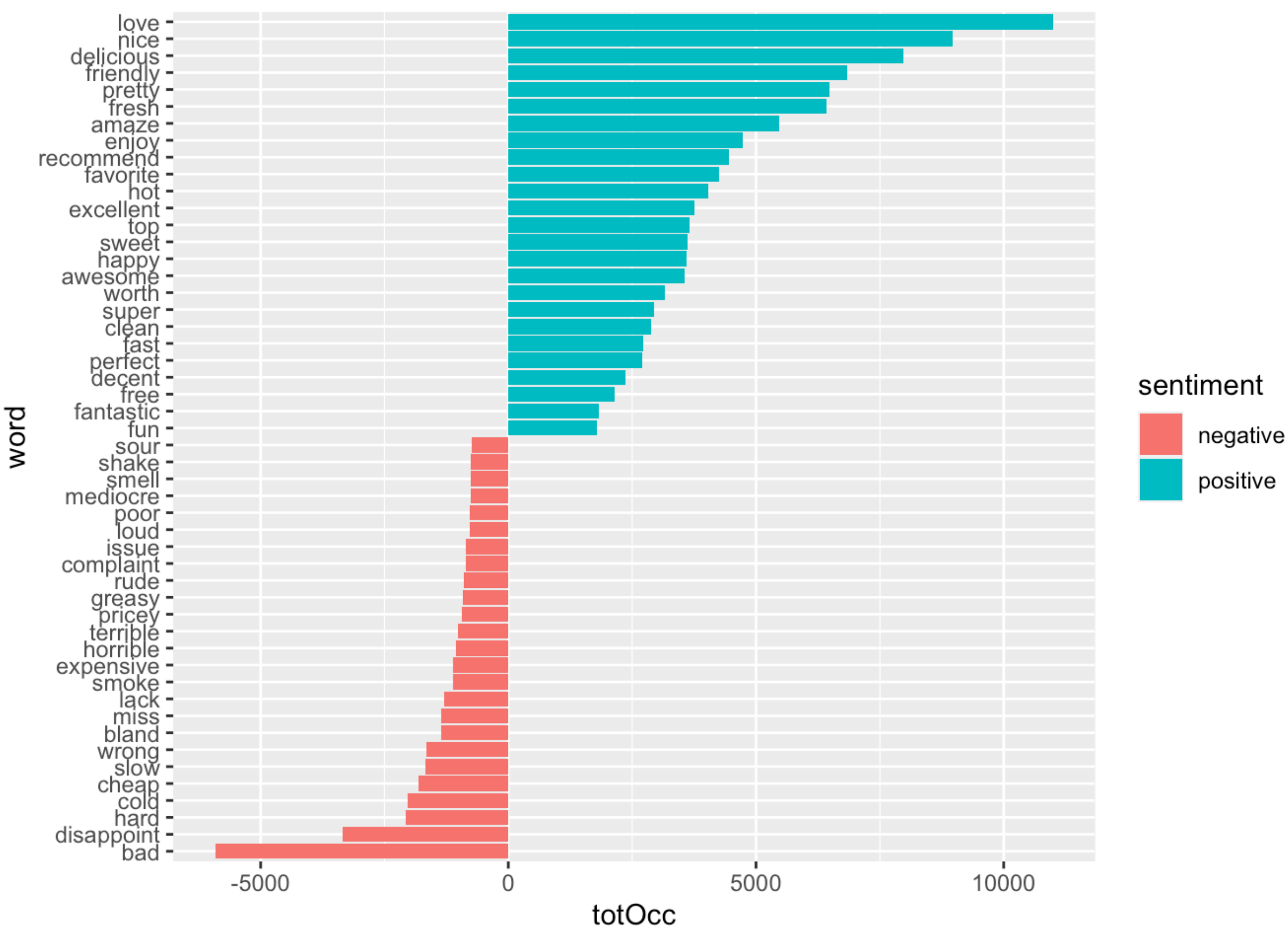


Using Bing Dictionary

Sentiment Analysis using the Bing dictionary gave us the words with either positive or negative sentiment. Then we summarized the sentiment words per review. Thereafter a sentiment score was calculated based on the proportion of the positive or negative words. Summarizing the entire analysis against the star ratings gave us the below table:

```
## Dictionary 1 - Bing
#Analyze Which words contribute to positive/negative sentiment - we can count the occ
urrences of positive/negative sentiment words in the reviews
xx = from_bing_dict %>% group_by(word, sentiment) %>% summarise(totOcc=sum(n)) %>% ar
range(sentiment, desc(totOcc))
#negate the counts for the negative sentiment words
xx = xx %>% mutate (totOcc=ifelse(sentiment=="positive", totOcc, -totOcc))
#the most positive and most negative words
# ungrouping is important because we have grouped by word and sentiment together in t
he code above
xx = ungroup(xx)
#top_n(xx, 25) %>% arrange(sentiment, desc(totOcc))
#top_n(xx, -25)  %>% arrange(sentiment, desc(totOcc))
orderw = rbind(top_n(xx, 25), top_n(xx, -25)) %>% mutate(word=reorder(word,totOcc))
# Review Sentiment Analysis
#summarise positive/negative sentiment words per review
rev_senti_bing = from_bing_dict %>% group_by(review_id, starsReview) %>% summarise(nw
ords=n(),posSum=sum(sentiment=='positive'), negSum=sum(sentiment=='negative'))
```

```
ggplot(orderw,aes(word, totOcc, fill= sentiment)) +geom_col()+coord_flip() #SHOULD I
INCLUDE COLOR
```



```
#calculate sentiment score based on proportion of positive, negative words
rev_senti_bing = rev_senti_bing %>% mutate(posProp=posSum/nwords, negProp=negSum/nwor
ds)
rev_senti_bing = rev_senti_bing %>% mutate(sentiScore=posProp-negProp)
rev_senti_bing %>% group_by(starsReview) %>% summarise(avgPos=mean(posProp), avgNeg=m
ean(negProp), avgSentiSc=mean(sentiScore))
```

```
## # A tibble: 5 × 4
##   starsReview avgPos avgNeg avgSentiSc
##         <int>  <dbl>  <dbl>      <dbl>
## 1           1  0.308  0.692     -0.384
## 2           2  0.447  0.553     -0.106
## 3           3  0.610  0.390      0.220
## 4           4  0.749  0.251      0.497
## 5           5  0.829  0.171      0.659
```

```
#considering reviews with 1 & 2 starsReview as negative, and this with 4 & 5 starsRev
iew as positive
rev_senti_bing = rev_senti_bing %>% mutate(hiLo=ifelse(starsReview<=2,-1, ifelse(star
sReview>=4, 1, 0 )))
rev_senti_bing = rev_senti_bing %>% mutate(pred_hiLo=ifelse(sentiScore >0, 1, -1))
xx_bing = rev_senti_bing %>% filter(hiLo!=0)
confusion_matrix_bing = table(actual=xx_bing$hiLo, predicted=xx_bing$pred_hiLo )
```

We took only those words which match the BING dictionary. The average positive score in the above table is nothing but the mean of the positive proportion and the average negative score is the mean of the negative proportion. Sentiment score for a single review is the difference between positive and negative proportions. Average sentiment score is the mean of all sentiment scores grouped by the star rating.

```
#calculating the accuracy of the predictions
confusionMatrix(confusion_matrix_bing)
```

```
## Confusion Matrix and Statistics
##
##        predicted
## actual    -1     1
##     -1  7304  2326
##     1   3900 24018
##
##                Accuracy : 0.8342
##                  95% CI : (0.8304, 0.8379)
##     No Information Rate : 0.7016
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5873
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6519
##             Specificity : 0.9117
##          Pos Pred Value : 0.7585
##          Neg Pred Value : 0.8603
##              Prevalence : 0.2984
##          Detection Rate : 0.1945
##    Detection Prevalence : 0.2565
##       Balanced Accuracy : 0.7818
##
##        'Positive' Class : -1
##
```

Using NRC Dictionary

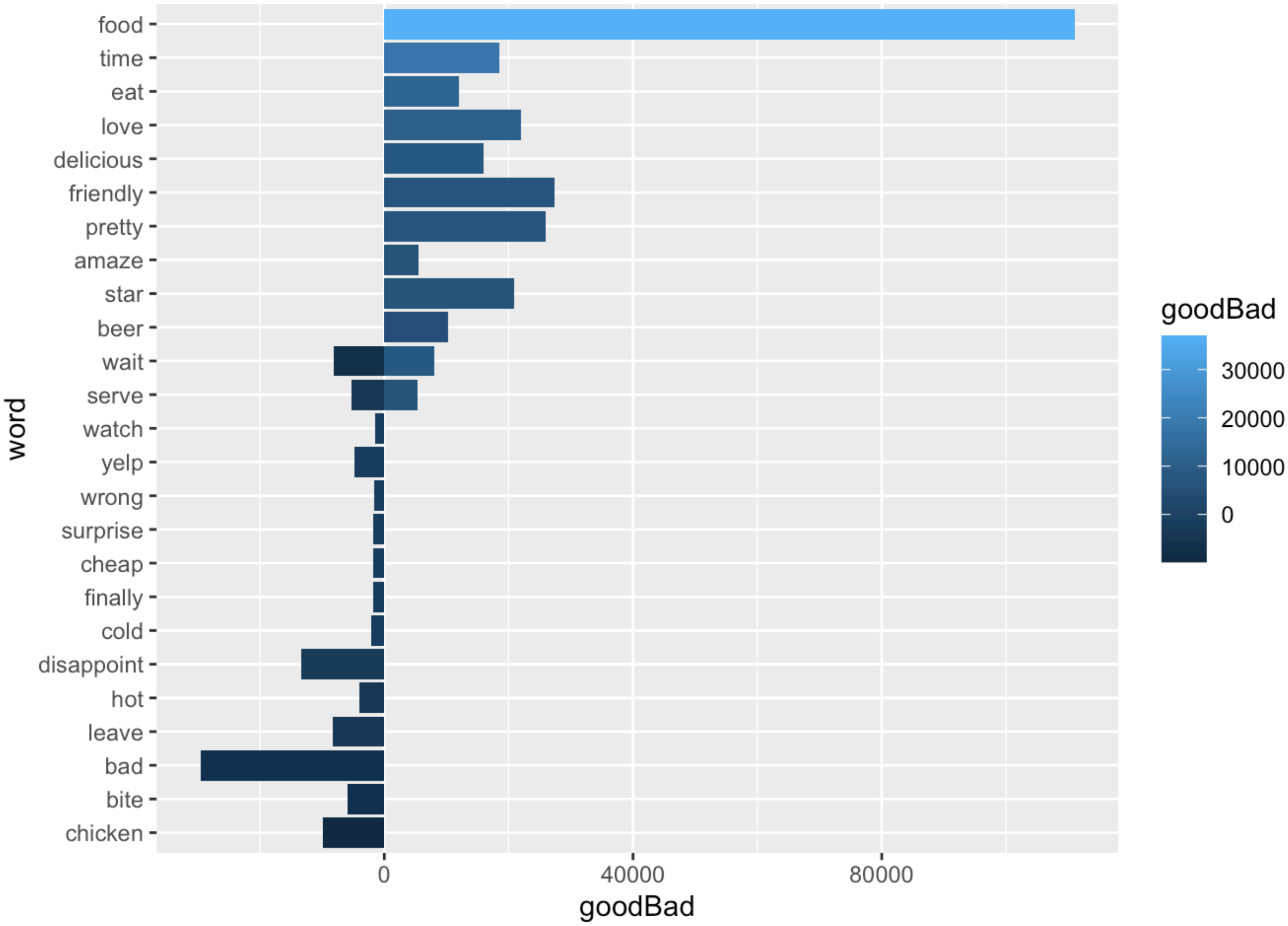
Sentiment Analysis using NRC gave us several sentiment categories - all the words were grouped into one of these categories. Considering {anger, disgust, fear, sadness, negative} to denote 'bad' reviews, and {positive, joy, anticipation, trust, surprise} to denote 'good' reviews we got the GoodBad score for each word.

```
## Dictionary 2 - NRC
senti_nrc = from_nrc_dict %>% group_by (word, sentiment) %>% summarise(totOcc=sum(n))
%>% arrange(sentiment, desc(totOcc))
#top few words for different sentiments
senti_nrc %>% group_by(sentiment) %>% arrange(sentiment, desc(totOcc)) %>% top_n(10)
```

```
## # A tibble: 100 × 3
## # Groups:   sentiment [10]
##    word       sentiment totOcc
##    <chr>      <chr>      <int>
##  1 bad        anger       5918
##  2 hot        anger       4041
##  3 disappoint anger       3337
##  4 yelp       anger       1590
##  5 buffet     anger       1361
##  6 money      anger       1333
##  7 hit        anger       1150
##  8 horrible   anger       1063
##  9 excite     anger       1016
## 10 terrible   anger       1013
## # … with 90 more rows
```

```
# we have got total 10 sentiments
#considering  {anger, disgust, fear sadness, negative} to denote 'bad' reviews, and {
positive, joy, anticipation, trust,surprise} to denote 'good' reviews
# Geting the GoodBad score for each word
xx1 = senti_nrc %>% mutate(goodBad=ifelse(sentiment %in% c('anger', 'disgust', 'fear'
, 'sadness', 'negative'), -totOcc, ifelse(sentiment %in% c('positive', 'joy', 'antici
pation', 'trust','surprise'), totOcc, 0)))
xx1 = ungroup(xx1)

nrcwords = rbind(top_n(xx1, 25), top_n(xx1, -25)) %>% mutate(word=reorder(word,goodBa
d))
##Analysis by Review
rev_senti_nrc = from_nrc_dict %>% group_by (review_id, starsReview, sentiment) %>% su
mmarise(totOcc=sum(n)) %>% arrange(starsReview, sentiment, desc(totOcc))
## Geting the GoodBad score for each review
xx2 = rev_senti_nrc %>% mutate(goodBad=ifelse(sentiment %in% c('anger', 'disgust', 'f
ear', 'sadness', 'negative'), -totOcc, ifelse(sentiment %in% c('positive', 'joy', 'an
ticipation', 'trust','surprise'), totOcc, 0)))
xx2 = ungroup(xx2)

rev_senti_nrc = xx2 %>% group_by(review_id, starsReview) %>% summarise(nwords=n(),sen
tiGoodBad =sum(goodBad))
```

```
ggplot(nrcwords,aes(word, goodBad, fill=goodBad)) +geom_col()+coord_flip()
```




```
rev_senti_nrc %>% group_by(starsReview)%>%summarise(avgLen=mean(nwords),avgSenti=mean
(sentiGoodBad))
```

```
## # A tibble: 5 × 3
##   starsReview avgLen avgSenti
##         <int>  <dbl>    <dbl>
## 1           1   8.11     4.23
## 2           2   7.86     9.43
## 3           3   7.37    12.7
## 4           4   6.81    14.4
## 5           5   6.26    13.6
```

```
#considering reviews with 1 & 2 starsReview as negative, and this with 4 & 5 starsRev
iew as positive
rev_senti_nrc = rev_senti_nrc %>% mutate(hiLo=ifelse(starsReview<=2,-1, ifelse(starsR
eview>=4, 1, 0 )))
rev_senti_nrc = rev_senti_nrc %>% mutate(pred_hiLo=ifelse(sentiGoodBad >0, 1, -1))
xx_nrc1 = rev_senti_nrc %>% filter(hiLo!=0)
confusion_matrix_nrc = table(actual=xx_nrc1$hiLo, predicted=xx_nrc1$pred_hiLo )
```

Question 4

Consider a basic approach (not developing a predictive model like a decision tree, random forests etc.) to use the dictionary based positive and negative terms to predict sentiment (positive or negative based on star rating) of a review. One approach for this is: based on each dictionary, obtain an aggregated positiveScore and a negativeScore for each review; for the AFINN dictionary, an aggregate positivity score can be obtained for each review.

a. Describe how you obtain the aggregated scores, and predictions based on these scores

b. What is the performance of this approach (for each dictionary). Does any dictionary perform better?

```
#calculating the accuracy of the predictions using afinn dictionary
#accuracy on training & test data
confusionMatrix(confusion_matrix_nrc)
```

```
## Confusion Matrix and Statistics
##
##       predicted
## actual    -1     1
##     -1  3067  6819
##     1   1539 26895
##
##                Accuracy : 0.7819
##                  95% CI : (0.7777, 0.786)
##     No Information Rate : 0.8798
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3101
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.66587
##             Specificity : 0.79774
##          Pos Pred Value : 0.31024
##          Neg Pred Value : 0.94587
##              Prevalence : 0.12020
##          Detection Rate : 0.08004
##    Detection Prevalence : 0.25799
##       Balanced Accuracy : 0.73181
##
##        'Positive' Class : -1
##
```

Using AFINN Dictionary

Analysis by Review Sentiment using Affin Dictionary gave the aggregate sentiment score according to to each star rating as follows:

1 star is lower rated and its average sentiment is negative 2.41 whereas 5 star is the highest rating with an average sentiment score of 7.28. It is worth noting that the average length is approximately the same for all ratings, as we would expect. Word length is really not the criteria which impacts the review sentiment.

However the aggregated scores help us to predict the review sentiment as we have seen above. Higher scores mean better review sentiment.

```
# With Dictionary 3 -  Afin - Review Sentiment Analysis
#Analysis by Review Sentiment
rev_senti_afinn = from_afin_dict %>% group_by(review_id, starsReview) %>% summarise(n
words=n(), sentiSum =sum(value))
```

```
rev_senti_afinn %>% group_by(starsReview) %>% summarise(avgLen=mean(nwords),avgSenti=
mean(sentiSum))
```

```
## # A tibble: 5 × 3
##   starsReview avgLen avgSenti
##         <int>  <dbl>    <dbl>
## 1           1   4.92   -2.42
## 2           2   5.13    0.881
## 3           3   4.94    3.76
## 4           4   4.84    6.44
## 5           5   4.39    7.28
```

```
#considering reviews with 1 & 2 starsReview as negative, and this with 4 & 5 starsRev
iew as positive
rev_senti_afinn = rev_senti_afinn %>% mutate(hiLo=ifelse(starsReview<=2,-1, ifelse(st
arsReview>=4, 1, 0 )))
rev_senti_afinn = rev_senti_afinn %>% mutate(pred_hiLo=ifelse(sentiSum >0, 1, -1))
xx<-rev_senti_afinn %>% filter(hiLo!=0)
confusion_matrix_afinn = table(actual=xx$hiLo, predicted=xx$pred_hiLo )
```

```
#calculating the accuracy of the predictions using afin dictionary
#accuracy on training & test data
confusionMatrix(confusion_matrix_afinn)
```

```
## Confusion Matrix and Statistics
##
##       predicted
## actual    -1     1
##     -1  5804  3597
##     1   2396 24933
##
##                Accuracy : 0.8368
##                  95% CI : (0.833, 0.8406)
##     No Information Rate : 0.7767
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5529
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7078
##             Specificity : 0.8739
##          Pos Pred Value : 0.6174
##          Neg Pred Value : 0.9123
##              Prevalence : 0.2233
##          Detection Rate : 0.1580
##    Detection Prevalence : 0.2559
##       Balanced Accuracy : 0.7909
##
##        'Positive' Class : -1
##
```

Below table shows prediction accuracy using different dictionaries:

```
col1 = c('BING','NRC','AFINN')
col2 = c(83.42,78.19,83.68)
colna = c('Dictionary','Accuracy (%)')
metric_df = data.frame(cbind(col1,col2))
names(metric_df) = colna
metric_df
```

```
##   Dictionary Accuracy (%)
## 1       BING        83.42
## 2        NRC        78.19
## 3      AFINN        83.68
```

Question 5

5. Develop models to predict review sentiment. For this, split the data randomly into training and test sets. To make run times manageable, you may take a smaller sample of reviews (minimum should be 10,000). You should consider models built using only the terms matching the sentiment dictionaries, as well as by using a broader list of terms (the idea here being, maybe words other than only the dictionary terms

can be useful). You should develop at least three different types of models (Naïve Bayes, and at least two others of your choice ….Lasso logistic regression (why Lasso?), xgb, random forest (use ranger for faster run-times) – use the same three modeling techniques with each of the dictionaries, with the combination of dictionary terms, and wit the broader set of terms.

(a)How do you evaluate performance? Which performance measures do you use, why?

(b)Which types of models does your team choose to develop, and why? Do you use term frequency, tfidf, or other measures, and why?

c. Develop models using only the sentiment dictionary terms – try the three different dictionaries; how do the dictionaries compare in terms of predictive performance? Then with a combination of the three dictionaries, ie. combine all dictionary terms. What is the size of the document-term matrix? Should you use stemming or lemmatization when using the dictionaries? Why?

d. Develop models using a broader list of terms (i.e. not restricted to the dictionary terms only) – how do you obtain these terms? Will you use stemming or lemmatization here, and why?

e. Compare performance of the models. How does performance here relate to that from Question 4 above. Explain your findings (and is this what you expected).

Prediction models using each dictionary

We did random sampling and have taken sampled data to run models. Also split the training and test data into 50-50 to avoid computation power issues. Before building prediction models, We removed reviews with rating 3 (neutral sentiment), since we are building a binary classification model and want to classify reviews as either as positive or negative sentiment.

Random Forest for Bing Dictionary

After Removing Star rating 3 from the data set we have 37548 rows. The table below shows the distribution of bing dictionary's rating for words in our data set.

9630 words have bing sentiment rating as -1 which means they have Star rating 1,2. Whereas we have 27918 words with bing sentiment rating as 1 which means they have Star rating 4 and 5

```
senti_bing_data = from_bing_dict %>%  pivot_wider(id_cols = c(review_id,starsReview),
names_from = word, values_from = tf_idf)  %>% ungroup()
#filter out the reviews with starsReview=3, and calculate hiLo sentiment 'class'
senti_bing_data = senti_bing_data %>% filter(starsReview!=3) %>% mutate(hiLo=ifelse(s
tarsReview<=2, -1, 1)) %>% select(-starsReview)
```

```
#how many review with 1, -1  'class'
senti_bing_data %>% group_by(hiLo) %>% tally()
```

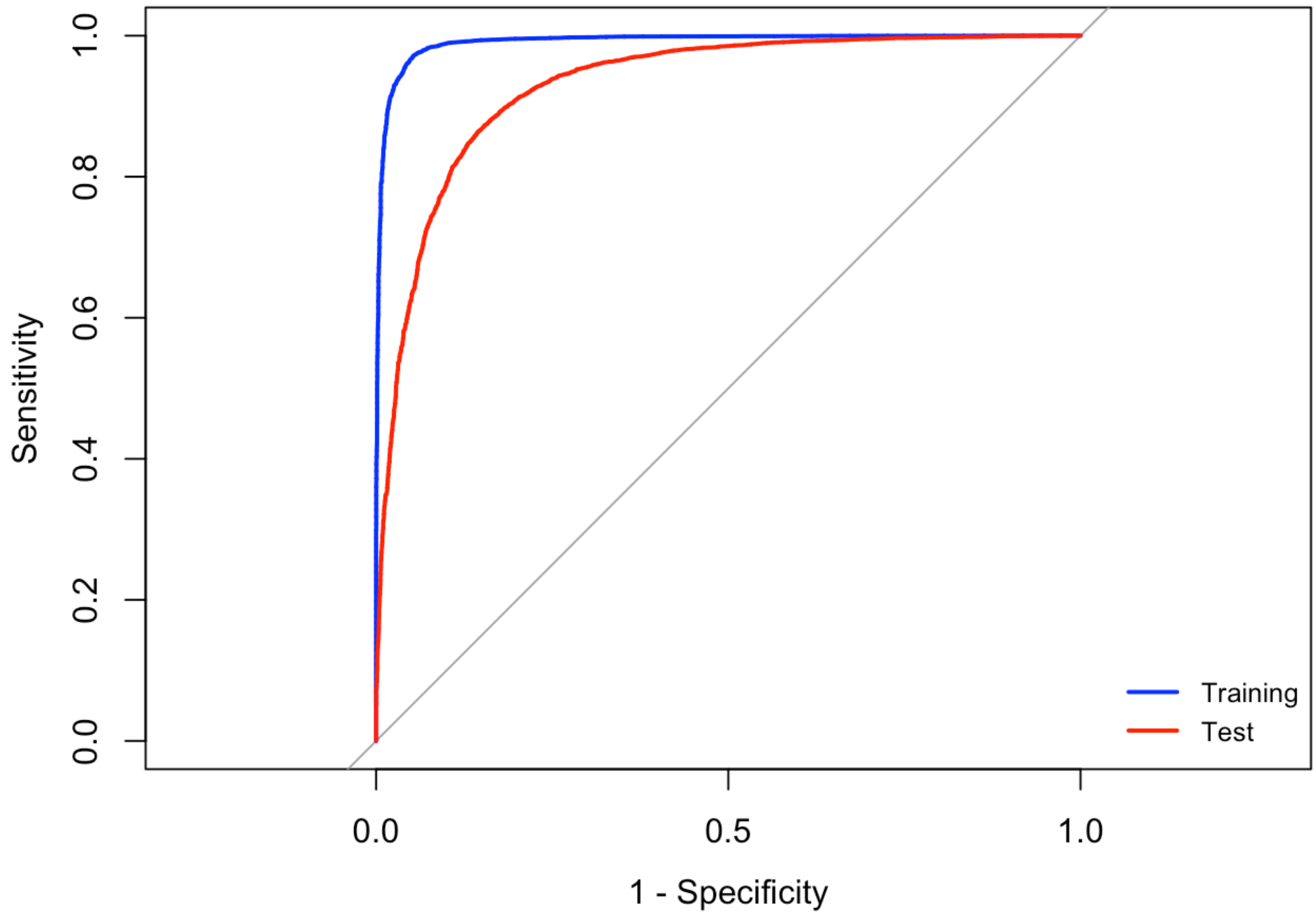
```
## # A tibble: 2 × 2
##    hiLo     n
##   <dbl> <int>
## 1    -1  9630
## 2     1 27918
```

```
#replace all the NAs with 0
senti_bing_data = senti_bing_data %>% replace(., is.na(.), 0)
senti_bing_data$hiLo = as.factor(senti_bing_data$hiLo)
#Create Dataset of 37,000 records
set.seed(213)
senti_bing_data_37k = senti_bing_data[sample(nrow(senti_bing_data),37000),]
set.seed(213)
senti_bing_data_37k_split = initial_split(senti_bing_data_37k, 0.5)
senti_bing_data_37k_trn = training(senti_bing_data_37k_split)
senti_bing_data_37k_tst = testing(senti_bing_data_37k_split)

set.seed(213)
rev_senti_bing_dat = rev_senti_bing[sample(nrow(rev_senti_bing),40000),]
set.seed(213)
rev_senti_bing_dat_split = initial_split(rev_senti_bing_dat, 0.5)
rev_senti_bing_dat_trn = training(rev_senti_bing_dat_split)
rev_senti_bing_dat_tst = testing(rev_senti_bing_dat_split)
```

```
## bing - ranger
rfModel_bing = ranger(dependent.variable.name = "hiLo", data=senti_bing_data_37k_trn
%>% select(-review_id), num.trees = 200, importance = 'permutation', probability = TR
UE)
```

```
## Computing permutation importance.. Progress: 17%. Estimated remaining time: 2 minu
tes, 52 seconds.
## Computing permutation importance.. Progress: 37%. Estimated remaining time: 1 minu
te, 58 seconds.
## Computing permutation importance.. Progress: 56%. Estimated remaining time: 1 minu
te, 18 seconds.
## Computing permutation importance.. Progress: 74%. Estimated remaining time: 47 sec
onds.
## Computing permutation importance.. Progress: 92%. Estimated remaining time: 14 sec
onds.
```

```
#Obtain predictions, and calculate performance
bing_rf_trn_preds = predict(rfModel_bing, senti_bing_data_37k_trn %>% select(-review_
id))$predictions
bing_rf_tst_preds = predict(rfModel_bing, senti_bing_data_37k_tst %>% select(-review_
id))$predictions
#The optimal threshold from the ROC analyses
rocTrn = roc(senti_bing_data_37k_trn$hiLo, bing_rf_trn_preds[,2], levels=c(-1, 1))
rocTst = roc(senti_bing_data_37k_tst$hiLo, bing_rf_tst_preds[,2], levels=c(-1, 1))
plot.roc(rocTrn, col='blue', legacy.axes = TRUE)
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("blue", "red"), lwd=2, cex=0.8, bty='n')
```



Naive-Bayes for Bing Dictionary

```
nb_Bing <- naiveBayes(hiLo ~ ., data=rev_senti_bing %>% select(-review_id))
rev_senti_Bing_NBTrn <- predict(nb_Bing, rev_senti_bing_dat_trn, type = "raw")
rev_senti_Bing_NBTst <- predict(nb_Bing, rev_senti_bing_dat_tst, type = "raw")
table(actual= rev_senti_bing_dat_trn$hiLo, predicted= rev_senti_Bing_NBTrn[,2]>0.5)
```

```
##       predicted
## actual FALSE  TRUE
##     -1  3993   392
##     0      0  2826
##     1  11932   857
```

```
table(actual= rev_senti_bing_dat_tst$hiLo, predicted= rev_senti_Bing_NBTst[,2]>0.5)
```

```
##       predicted
## actual FALSE  TRUE
##     -1  4003   417
##     0      0  2843
##     1  11899   838
```

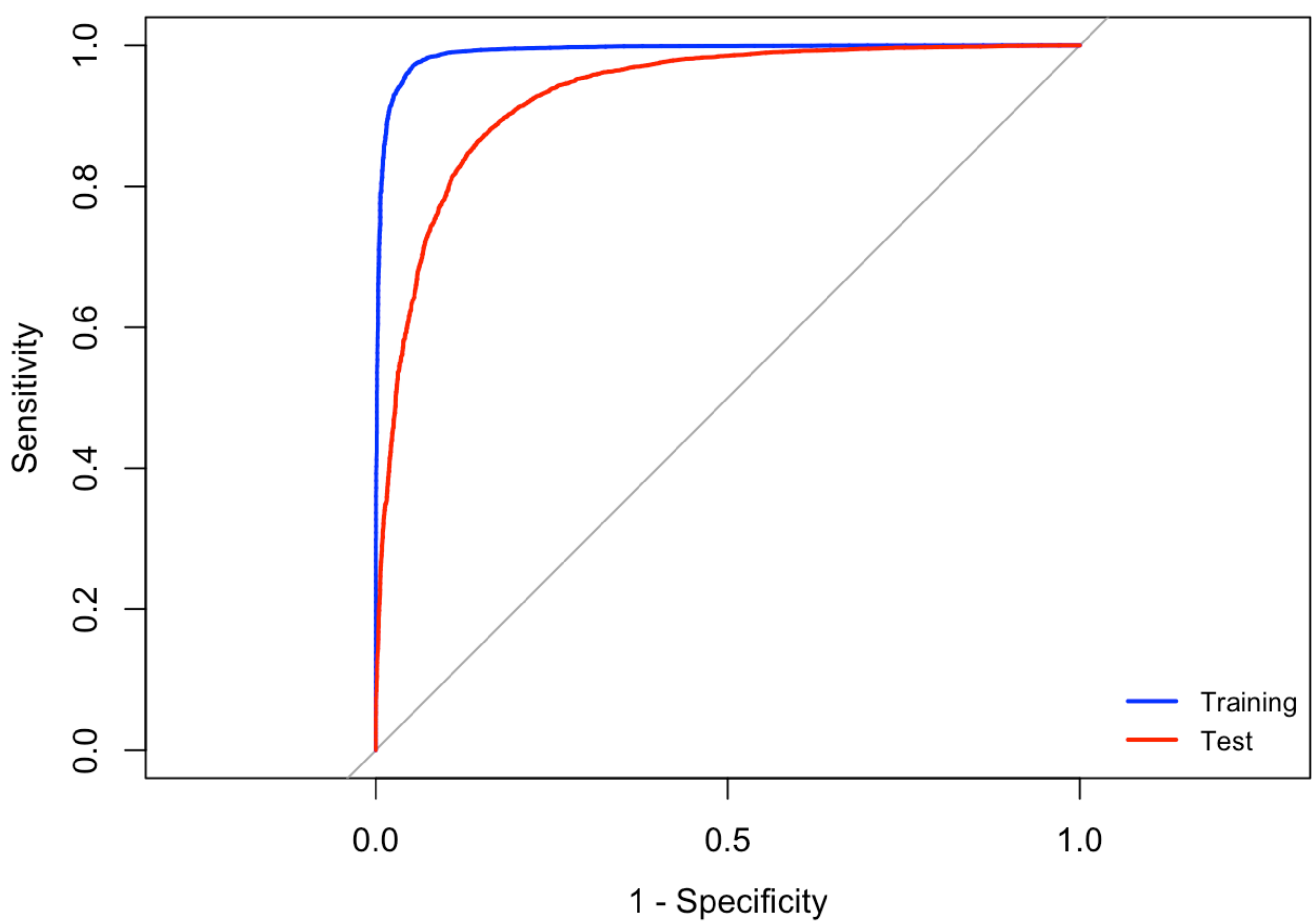
```
auc(as.numeric(rev_senti_bing_dat_trn$hiLo), rev_senti_Bing_NBTrn[,2])
```

```
## Area under the curve: 0.9993
```

```
auc(as.numeric(rev_senti_bing_dat_tst$hiLo), rev_senti_Bing_NBTst[,2])
```

```
## Area under the curve: 0.999
```

```
rocTrn_nb = roc(rev_senti_bing_dat_trn$hiLo, rev_senti_Bing_NBTrn[,2], levels=c(-1, 1
))
rocTst_nb = roc(rev_senti_bing_dat_tst$hiLo, rev_senti_Bing_NBTst[,2], levels=c(-1, 1
))
plot.roc(rocTrn, col='blue', legacy.axes = TRUE)
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("blue", "red"), lwd=2, cex=0.8, bty='n')
```



```
confusion_matrix_bing_nb = table(actual=rev_senti_bing_dat_trn$hiLo, predicted=rev_se
nti_bing_dat_tst$hiLo )
confusionMatrix(confusion_matrix_bing_nb)
```

```
## Confusion Matrix and Statistics
##
##       predicted
## actual   -1    0    1
##     -1  954  653 2778
##     0   625  392 1809
##     1  2841 1798 8150
##
## Overall Statistics
##
##                Accuracy : 0.4748
##                  95% CI : (0.4679, 0.4817)
##     No Information Rate : 0.6368
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : -0.0019
##
##  Mcnemar's Test P-Value : 0.7165
##
## Statistics by Class:
##
##                      Class: -1 Class: 0 Class: 1
## Sensitivity             0.2158   0.1379   0.6399
## Specificity             0.7798   0.8581   0.3613
## Pos Pred Value          0.2176   0.1387   0.6373
## Neg Pred Value          0.7780   0.8573   0.3639
## Prevalence              0.2210   0.1421   0.6369
## Detection Rate          0.0477   0.0196   0.4075
## Detection Prevalence    0.2193   0.1413   0.6394
## Balanced Accuracy       0.4978   0.4980   0.5006
```

```
#Best threshold from ROC analyses
bThr = coords(rocTrn, "best", ret="threshold", transpose = FALSE)
bThr = as.numeric(bThr)
bThr
```

```
## [1] 0.6092488
```

```
#Confusion Matrix at bThr for Trn and Tst dataset
confusionMatrix(table(actual=senti_bing_data_37k_trn$hiLo, preds=if_else(bing_rf_trn_
preds[,2]>bThr,1,-1)))
```

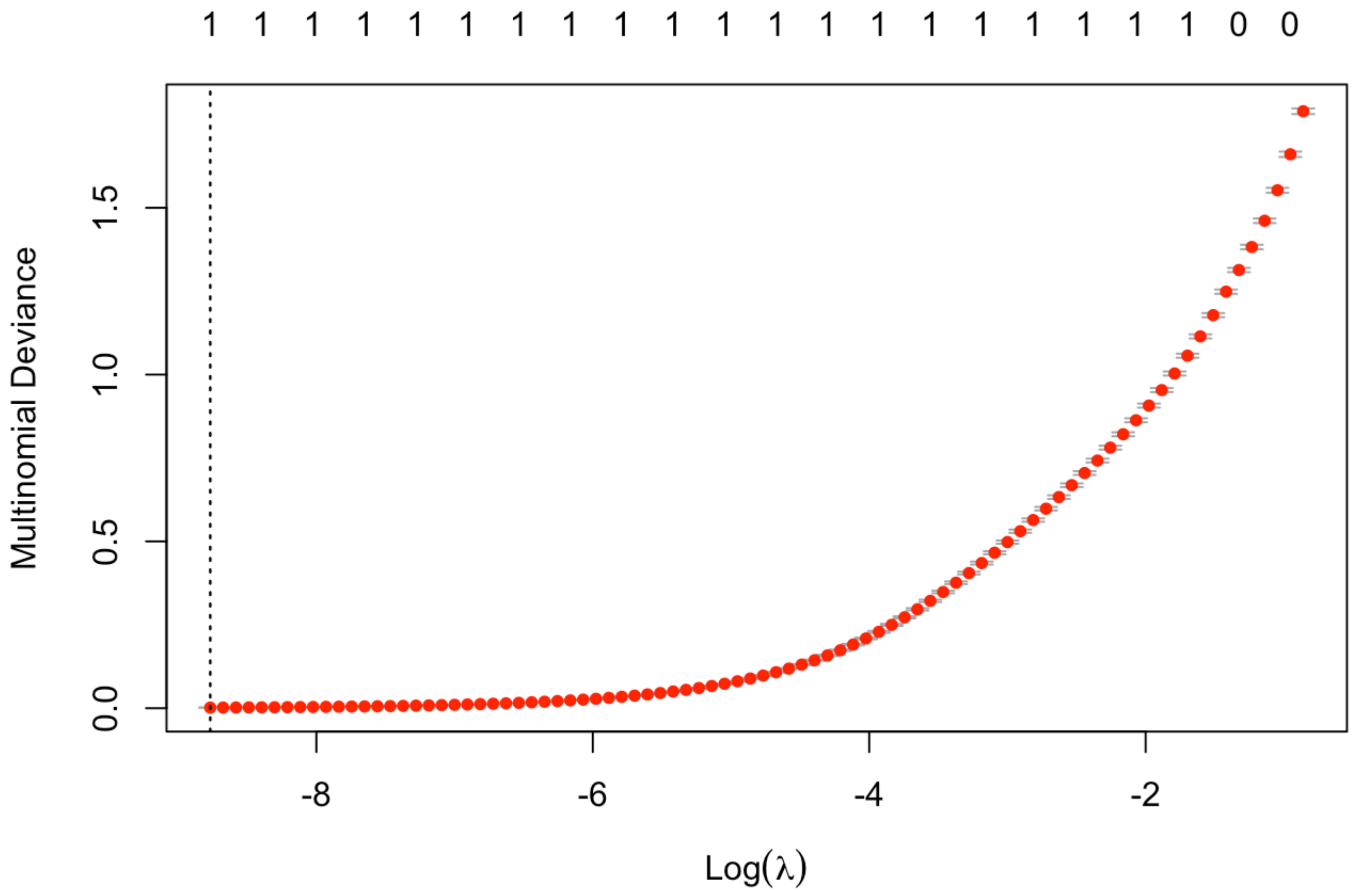
```
## Confusion Matrix and Statistics
##
##       preds
## actual    -1     1
##     -1  4534   252
##     1    392 13322
##
##                Accuracy : 0.9652
##                  95% CI : (0.9624, 0.9678)
##     No Information Rate : 0.7337
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9101
##
##  Mcnemar's Test P-Value : 4.317e-08
##
##             Sensitivity : 0.9204
##             Specificity : 0.9814
##          Pos Pred Value : 0.9473
##          Neg Pred Value : 0.9714
##              Prevalence : 0.2663
##          Detection Rate : 0.2451
##    Detection Prevalence : 0.2587
##       Balanced Accuracy : 0.9509
##
##        'Positive' Class : -1
##
```

```
confusionMatrix(table(actual=senti_bing_data_37k_tst$hiLo, preds=if_else(bing_rf_tst_
preds[,2]>bThr,1,-1)))
```

```
## Confusion Matrix and Statistics
##
##       preds
## actual    -1     1
##     -1  3687  1029
##     1   1092 12692
##
##                Accuracy : 0.8854
##                  95% CI : (0.8807, 0.8899)
##     No Information Rate : 0.7417
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6995
##
##  Mcnemar's Test P-Value : 0.1782
##
##             Sensitivity : 0.7715
##             Specificity : 0.9250
##          Pos Pred Value : 0.7818
##          Neg Pred Value : 0.9208
##              Prevalence : 0.2583
##          Detection Rate : 0.1993
##    Detection Prevalence : 0.2549
##       Balanced Accuracy : 0.8483
##
##        'Positive' Class : -1
##
```

Logistic Regression for Bing Dictionary

```
library(glmnet)
Log_Reg_Bing_x <- rev_senti_bing_dat_trn %>% select(-hiLo,-review_id)
Log_Reg_Bing_y <- rev_senti_bing_dat_trn$hiLo

Log_Reg_Bing_x_Tst <- rev_senti_bing_dat_tst %>% select(-hiLo,,-review_id)
Log_Reg_Bing_y_Tst <- rev_senti_bing_dat_tst$hiLo


set.seed(231)
cvglmnet_com <- cv.glmnet(data.matrix(Log_Reg_Bing_x), Log_Reg_Bing_y,
                        family = "multinomial",
                        nfolds = 5,
                        alpha = 1)
plot(cvglmnet_com)
```




```
########## variable importance glmnet
library(vip)
tb1 <- vi_model(cvglmnet_com)
arrange(tb1,desc(Importance),Variable)
```

```
## # A tibble: 9 × 3
##   Variable    Importance Sign
##   <chr>            <dbl> <chr>
## 1 starsReview       12.6 NEG
## 2 negProp            0   NEG
## 3 negSum             0   NEG
## 4 nwords             0   NEG
## 5 posProp            0   NEG
## 6 posSum             0   NEG
## 7 pred_hiLo          0   NEG
## 8 review_id          0   NEG
## 9 sentiScore         0   NEG
```

```r
sort(tb1$Importance, decreasing = TRUE) %>% view()

library(caret)
#confusion_matrix for Trn
glm_com_train_pred <- predict(cvglmnet_com, data.matrix(Log_Reg_Bing_x),s=cvglmnet_co
m$lambda.1se,type="class")
glm_com_train_pred <- factor(glm_com_train_pred, levels=c(1,-1))
Log_Reg_Bing_y_Tst2 <- factor(Log_Reg_Bing_y, levels=c(1,-1))
confusionMatrix (glm_com_train_pred,Log_Reg_Bing_y_Tst2, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1    -1
##         1  12789     0
##         -1     0  4385
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.7447
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7447
##          Detection Rate : 0.7447
##    Detection Prevalence : 0.7447
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 1
##
```

```r
#confusion_matrix for Tst
glm_com_test_pred <- predict(cvglmnet_com, data.matrix(Log_Reg_Bing_x_Tst),s=cvglmnet
_com$lambda.1se,type="class")
glm_com_test_pred <- factor(glm_com_test_pred, levels=c(1,-1))
Log_Reg_Bing_y_Tst2 <- factor(Log_Reg_Bing_y_Tst, levels=c(1,-1))
confusionMatrix (glm_com_test_pred,Log_Reg_Bing_y_Tst2, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1    -1
##         1  12737     0
##         -1     0  4420
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.7424
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7424
##          Detection Rate : 0.7424
##    Detection Prevalence : 0.7424
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 1
##
```

Random Forest for NRC Dictionary

After Removing Star rating 3 from the data set we have 38320 rows. The table below shows the distribution of NRC dictionary's rating for words in our data set.

```
## nrc
from_nrc_dict_1 = from_nrc_dict[,-2]
from_nrc_dict_1 = from_nrc_dict_1[!duplicated(from_nrc_dict_1), ]
#create Document Term Matrix
senti_nrc_data = from_nrc_dict_1 %>%  pivot_wider(id_cols = c(review_id,starsReview),
names_from = word, values_from = tf_idf)  %>% ungroup()
senti_nrc_data = senti_nrc_data %>% filter(starsReview!=3) %>% mutate(hiLo=ifelse(sta
rsReview<=2, -1, 1)) %>% select(-starsReview)
senti_nrc_data = senti_nrc_data %>% replace(., is.na(.), 0)
senti_nrc_data$hiLo = as.factor(senti_nrc_data$hiLo)
senti_nrc_data %>% group_by(hiLo) %>% tally()
```

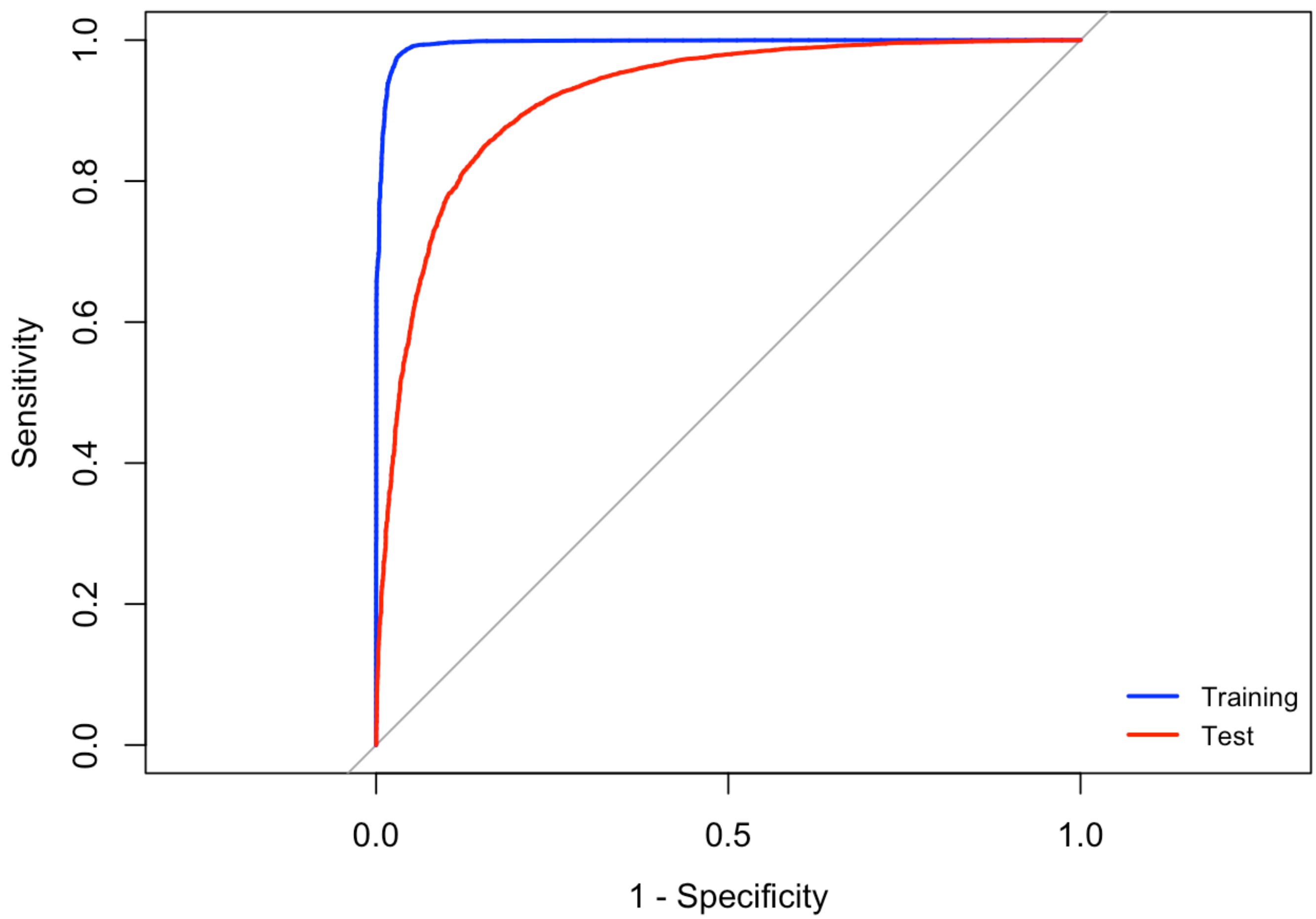
```
## # A tibble: 2 × 2
##   hiLo      n
##   <fct> <int>
## 1 -1     9886
## 2 1     28434
```

```
set.seed(213)
senti_nrc_data_38k = senti_nrc_data[sample(nrow(senti_nrc_data),38000),]
set.seed(213)
senti_nrc_data_38k_split = initial_split(senti_nrc_data_38k, 0.5)
senti_nrc_data_38k_trn = training(senti_nrc_data_38k_split)
senti_nrc_data_38k_tst = testing(senti_nrc_data_38k_split)

set.seed(213)
rev_senti_nrc_dat = senti_nrc_data[sample(nrow(senti_nrc_data),38000),]
set.seed(213)
rev_senti_nrc_dat_split = initial_split(rev_senti_nrc_dat, 0.5)
rev_senti_nrc_dat_trn = training(rev_senti_nrc_dat_split)
rev_senti_nrc_dat_tst = testing(rev_senti_nrc_dat_split)
```

```
## nrc - ranger
rfModel_nrc = ranger(dependent.variable.name = "hiLo", data=senti_nrc_data_38k_trn %>
% select(-review_id), num.trees = 200, importance ='permutation', probability = TRUE)
```

```
## Computing permutation importance.. Progress: 14%. Estimated remaining time: 3 minu
tes, 18 seconds.
## Computing permutation importance.. Progress: 28%. Estimated remaining time: 2 minu
tes, 35 seconds.
## Computing permutation importance.. Progress: 45%. Estimated remaining time: 1 minu
te, 56 seconds.
## Computing permutation importance.. Progress: 61%. Estimated remaining time: 1 minu
te, 21 seconds.
## Computing permutation importance.. Progress: 77%. Estimated remaining time: 49 sec
onds.
## Computing permutation importance.. Progress: 92%. Estimated remaining time: 16 sec
onds.
```

```
#Obtain predictions, and calculate performance
nrc_rf_trn_preds = predict(rfModel_nrc, senti_nrc_data_38k_trn %>% select(-review_id)
)$predictions
nrc_rf_tst_preds = predict(rfModel_nrc, senti_nrc_data_38k_tst %>% select(-review_id)
)$predictions
#The optimal threshold from the ROC analyses
rocTrn = roc(senti_nrc_data_38k_trn$hiLo, nrc_rf_trn_preds[,2], levels=c(-1, 1))
rocTst = roc(senti_nrc_data_38k_tst$hiLo, nrc_rf_tst_preds[,2], levels=c(-1, 1))
plot.roc(rocTrn, col='blue', legacy.axes = TRUE)
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),col=c("blue", "red"), lwd=2, cex=0
.8, bty='n')
```



Naive Bayes for NRC Dictionary

```
nb_nrc <- naiveBayes(hiLo ~ ., data=rev_senti_nrc_dat %>% select(-review_id))
rev_senti_nrc_NBTrn <- predict(nb_nrc, rev_senti_nrc_dat_trn, type = "raw")
rev_senti_nrc_NBTst <- predict(nb_nrc, rev_senti_nrc_dat_tst, type = "raw")
table(actual= rev_senti_nrc_dat_trn$hiLo, predicted= rev_senti_nrc_NBTrn[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1  3708 1254
##     1   6282 7756
```

```
table(actual= rev_senti_nrc_dat_tst$hiLo, predicted= rev_senti_nrc_NBTst[,2]>0.5)
```

```
##       predicted
## actual FALSE TRUE
##     -1  3692 1147
##     1   6397 7764
```

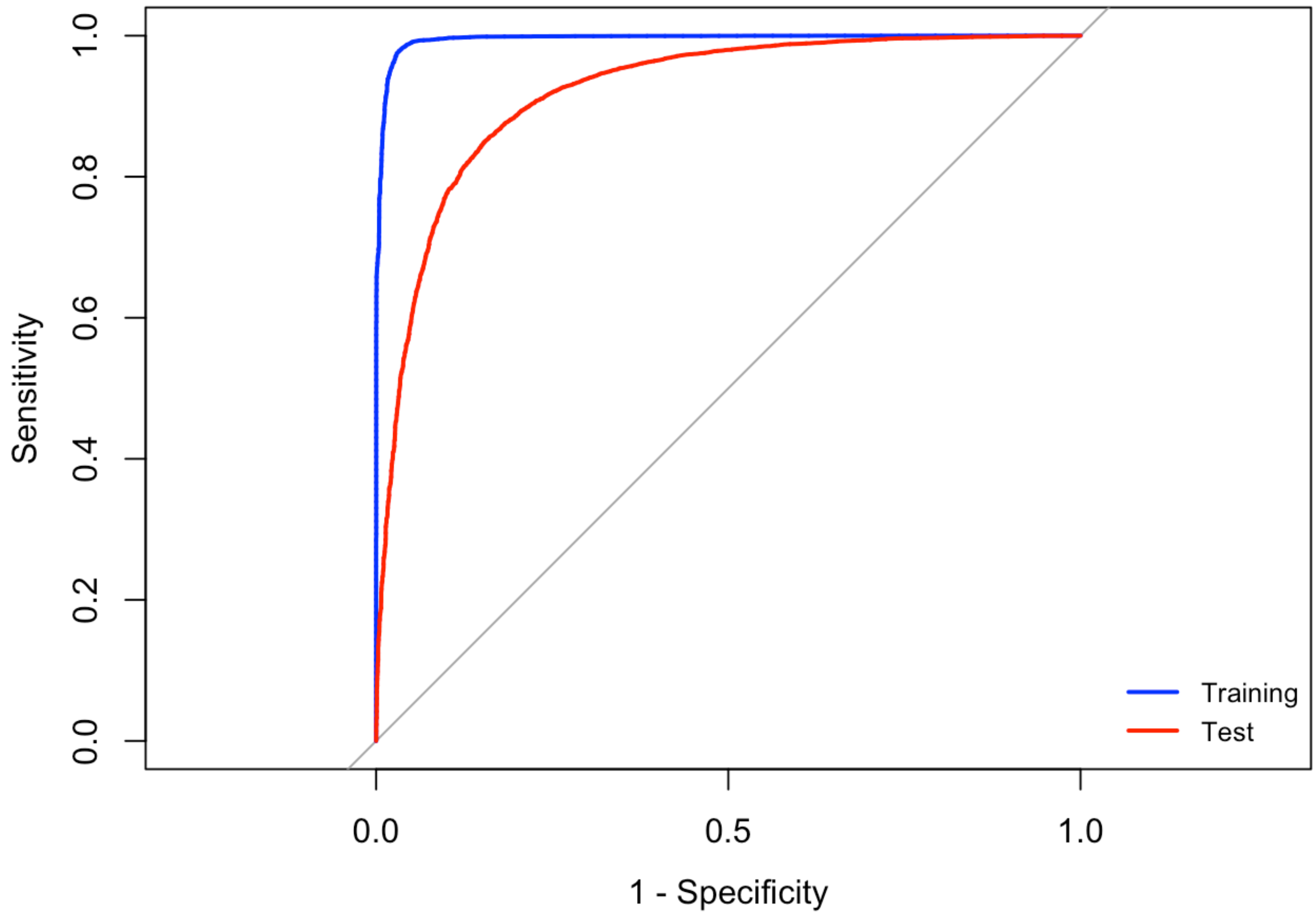
```
auc(as.numeric(rev_senti_nrc_dat_trn$hiLo), rev_senti_nrc_NBTrn[,2])
```

```
## Area under the curve: 0.7058
```

```
auc(as.numeric(rev_senti_nrc_dat_tst$hiLo), rev_senti_nrc_NBTst[,2])
```

```
## Area under the curve: 0.7094
```

```
rocTrn_nb = roc(rev_senti_nrc_dat_trn$hiLo, rev_senti_nrc_NBTrn[,2], levels=c(-1, 1))
rocTst_nb = roc(rev_senti_nrc_dat_tst$hiLo, rev_senti_nrc_NBTst[,2], levels=c(-1, 1))
plot.roc(rocTrn, col='blue', legacy.axes = TRUE)
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("blue", "red"), lwd=2, cex=0.8, bty='n')
```




```
confusion_matrix_nrc_nb = table(actual=rev_senti_nrc_dat_trn$hiLo, predicted=rev_senti_nrc_dat_tst$hiLo )
confusionMatrix(confusion_matrix_nrc_nb)
```

```
## Confusion Matrix and Statistics
##
##       predicted
## actual    -1     1
##     -1  1334  3628
##     1   3505 10533
##
##                Accuracy : 0.6246
##                  95% CI : (0.6176, 0.6315)
##     No Information Rate : 0.7453
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : 0.0193
##
##  Mcnemar's Test P-Value : 0.1486
##
##             Sensitivity : 0.27568
##             Specificity : 0.74380
##          Pos Pred Value : 0.26884
##          Neg Pred Value : 0.75032
##              Prevalence : 0.25468
##          Detection Rate : 0.07021
##    Detection Prevalence : 0.26116
##       Balanced Accuracy : 0.50974
##
##        'Positive' Class : -1
##
```

```
#Best threshold from ROC analyses
bThr = coords(rocTrn, "best", ret="threshold", transpose = FALSE)
bThr = as.numeric(bThr)
bThr
```

```
## [1] 0.6213443
```

```
#Confusion Matrix at bThr for Trn and Tst dataset
confusionMatrix(table(actual=senti_nrc_data_38k_trn$hiLo, preds=if_else(nrc_rf_trn_pr
eds[,2]>bThr,1,-1)))
```

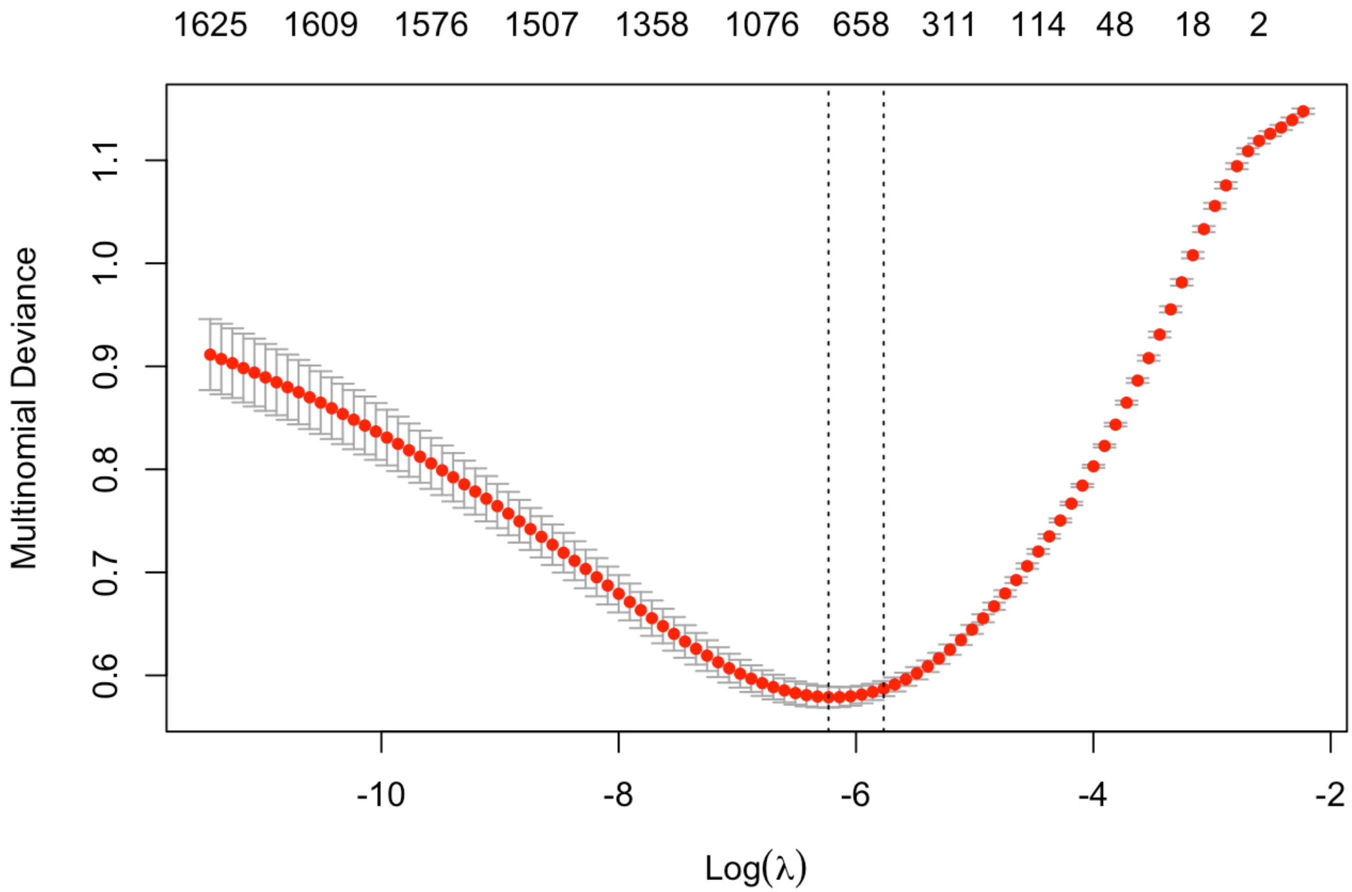
```
## Confusion Matrix and Statistics
##
##       preds
## actual    -1     1
##     -1  4799   163
##     1    298 13740
##
##                Accuracy : 0.9757
##                  95% CI : (0.9734, 0.9779)
##     No Information Rate : 0.7317
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9377
##
##  Mcnemar's Test P-Value : 4.348e-10
##
##             Sensitivity : 0.9415
##             Specificity : 0.9883
##          Pos Pred Value : 0.9672
##          Neg Pred Value : 0.9788
##              Prevalence : 0.2683
##          Detection Rate : 0.2526
##    Detection Prevalence : 0.2612
##       Balanced Accuracy : 0.9649
##
##        'Positive' Class : -1
##
```

```
confusionMatrix(table(actual=senti_nrc_data_38k_tst$hiLo, preds=if_else(nrc_rf_tst_pr
eds[,2]>bThr,1,-1)))
```

```
## Confusion Matrix and Statistics
##
##       preds
## actual    -1     1
##     -1  3775  1064
##     1   1384 12777
##
##                Accuracy : 0.8712
##                  95% CI : (0.8663, 0.8759)
##     No Information Rate : 0.7285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6679
##
##  Mcnemar's Test P-Value : 1.138e-10
##
##             Sensitivity : 0.7317
##             Specificity : 0.9231
##          Pos Pred Value : 0.7801
##          Neg Pred Value : 0.9023
##              Prevalence : 0.2715
##          Detection Rate : 0.1987
##    Detection Prevalence : 0.2547
##       Balanced Accuracy : 0.8274
##
##        'Positive' Class : -1
##
```

Logistic Regression for NRC Dictionary

```
library(glmnet)
Log_Reg_nrc_x <- rev_senti_nrc_dat_trn %>% select(-hiLo,-review_id)
Log_Reg_nrc_y <- rev_senti_nrc_dat_trn$hiLo

Log_Reg_nrc_x_Tst <- rev_senti_nrc_dat_tst %>% select(-hiLo,,-review_id)
Log_Reg_nrc_y_Tst <- rev_senti_nrc_dat_tst$hiLo


set.seed(231)
cvglmnet_com <- cv.glmnet(data.matrix(Log_Reg_nrc_x), Log_Reg_nrc_y,
                        family = "multinomial",
                        nfolds = 5,
                        alpha = 1)
plot(cvglmnet_com)
```




```r
########## variable importance glmnet
library(vip)
tb1 <- vi_model(cvglmnet_com)
arrange(tb1,desc(Importance),Variable)
```

```
## # A tibble: 1,639 × 3
##    Variable      Importance Sign
##    <chr>              <dbl> <chr>
##  1 delicious          11.4  NEG
##  2 perfect             7.42 NEG
##  3 apology             6.77 POS
##  4 horrible            6.72 POS
##  5 bland               6.71 POS
##  6 amaze               6.67 NEG
##  7 mediocre            6.52 POS
##  8 disgust             6.52 POS
##  9 uninteresting       6.09 POS
## 10 love                6.05 NEG
## # … with 1,629 more rows
```

```r
sort(tb1$Importance, decreasing = TRUE) %>% view()

library(caret)
#confusion_matrix for Trn
glm_com_train_pred <- predict(cvglmnet_com, data.matrix(Log_Reg_nrc_x),s=cvglmnet_com
$lambda.1se,type="class")
glm_com_train_pred <- factor(glm_com_train_pred, levels=c(1,-1))
Log_Reg_nrc_y_Tst2 <- factor(Log_Reg_nrc_y, levels=c(1,-1))
confusionMatrix (glm_com_train_pred,Log_Reg_nrc_y_Tst2, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1    -1
##         1  13623  1571
##         -1   415  3391
##
##                Accuracy : 0.8955
##                  95% CI : (0.891, 0.8998)
##     No Information Rate : 0.7388
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7071
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9704
##             Specificity : 0.6834
##          Pos Pred Value : 0.8966
##          Neg Pred Value : 0.8910
##              Prevalence : 0.7388
##          Detection Rate : 0.7170
##    Detection Prevalence : 0.7997
##       Balanced Accuracy : 0.8269
##
##        'Positive' Class : 1
##
```

```r
#confusion_matrix for Tst
glm_com_test_pred <- predict(cvglmnet_com, data.matrix(Log_Reg_nrc_x_Tst),s=cvglmnet_
com$lambda.1se,type="class")
glm_com_test_pred <- factor(glm_com_test_pred, levels=c(1,-1))
Log_Reg_nrc_y_Tst2 <- factor(Log_Reg_nrc_y_Tst, levels=c(1,-1))
confusionMatrix (glm_com_test_pred,Log_Reg_nrc_y_Tst2, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1    -1
##         1  13553  1637
##         -1   608  3202
##
##                Accuracy : 0.8818
##                  95% CI : (0.8772, 0.8864)
##     No Information Rate : 0.7453
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6653
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9571
##             Specificity : 0.6617
##          Pos Pred Value : 0.8922
##          Neg Pred Value : 0.8404
##              Prevalence : 0.7453
##          Detection Rate : 0.7133
##    Detection Prevalence : 0.7995
##       Balanced Accuracy : 0.8094
##
##        'Positive' Class : 1
##
```

Random Forest for AFINN Dictionary

After Removing Star rating 3 from the data set we have 36730 rows. The table below shows the distribution of AFINN dictionary's rating for words in our data set.

```
##afin
senti_afin_data = from_afin_dict %>%  pivot_wider(id_cols = c(review_id,starsReview),
names_from = word, values_from = tf_idf)  %>% ungroup()
#filter out the reviews with starsReview=3, and calculate hiLo sentiment 'class'
senti_afin_data = senti_afin_data %>% filter(starsReview!=3) %>% mutate(hiLo=ifelse(s
tarsReview<=2, -1, 1)) %>% select(-starsReview)
#how many review with 1, -1  'class'
senti_afin_data %>% group_by(hiLo) %>% tally()
```

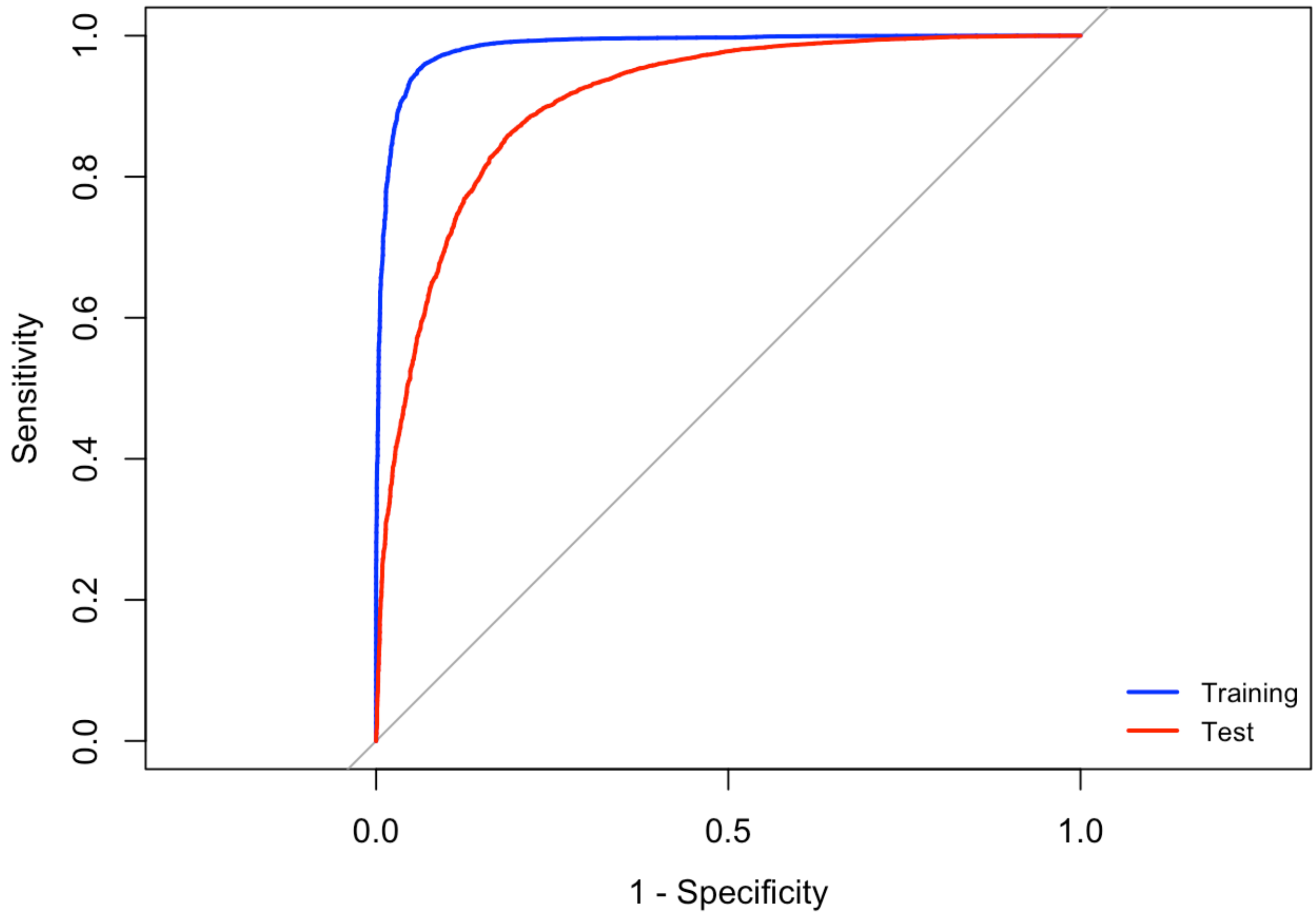
```
## # A tibble: 2 × 2
##    hiLo     n
##   <dbl> <int>
## 1    -1  9401
## 2     1 27329
```

```
#replace all the NAs with 0
senti_afin_data = senti_afin_data %>% replace(., is.na(.), 0)
senti_afin_data$hiLo = as.factor(senti_afin_data$hiLo)

set.seed(213)
senti_afin_data_36k = senti_afin_data[sample(nrow(senti_afin_data),36000),]
set.seed(213)
senti_afin_data_36k_split = initial_split(senti_afin_data_36k, 0.5)
senti_afin_data_36k_trn = training(senti_afin_data_36k_split)
senti_afin_data_36k_tst = testing(senti_afin_data_36k_split)

set.seed(213)
rev_senti_afin_dat = senti_afin_data[sample(nrow(senti_afin_data),36000),]
set.seed(213)
rev_senti_afin_dat_split = initial_split(rev_senti_afin_dat, 0.5)
rev_senti_afin_dat_trn = training(rev_senti_afin_dat_split)
rev_senti_afin_dat_tst = testing(rev_senti_afin_dat_split)
```

```
## afin - ranger
rfModel_afin = ranger(dependent.variable.name = "hiLo", data=senti_afin_data_36k_trn
%>% select(-review_id), num.trees = 200, importance='permutation', probability = TRUE
)
```

```
## Computing permutation importance.. Progress: 43%. Estimated remaining time: 41 sec
onds.
## Computing permutation importance.. Progress: 88%. Estimated remaining time: 8 seco
nds.
```

```
#Obtain predictions, and calculate performance
afin_rf_trn_preds = predict(rfModel_afin, senti_afin_data_36k_trn %>% select(-review_
id))$predictions
afin_rf_tst_preds = predict(rfModel_afin, senti_afin_data_36k_tst %>% select(-review_
id))$predictions
#The optimal threshold from the ROC analyses
rocTrn = roc(senti_afin_data_36k_trn$hiLo, afin_rf_trn_preds[,2], levels=c(-1, 1))
rocTst = roc(senti_afin_data_36k_tst$hiLo, afin_rf_tst_preds[,2], levels=c(-1, 1))
plot.roc(rocTrn, col='blue', legacy.axes = TRUE)
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("blue", "red"), lwd=2, cex=0.8, bty='n')
```




```
#Best threshold from ROC analyses
bThr = coords(rocTrn, "best", ret="threshold", transpose = FALSE)
bThr = as.numeric(bThr)
bThr
```

```
## [1] 0.6547677
```

```
#Confusion Matrix at bThr for Trn and Tst dataset
confusionMatrix(table(actual=senti_afin_data_36k_trn$hiLo, preds=if_else(afin_rf_trn_
preds[,2]>bThr,1,-1)))
```

```
## Confusion Matrix and Statistics
##
##       preds
## actual    -1     1
##     -1  4320   273
##     1    656 12751
##
##                Accuracy : 0.9484
##                  95% CI : (0.9451, 0.9516)
##     No Information Rate : 0.7236
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8678
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8682
##             Specificity : 0.9790
##          Pos Pred Value : 0.9406
##          Neg Pred Value : 0.9511
##              Prevalence : 0.2764
##          Detection Rate : 0.2400
##    Detection Prevalence : 0.2552
##       Balanced Accuracy : 0.9236
##
##        'Positive' Class : -1
##
```

```
confusionMatrix(table(actual=senti_afin_data_36k_tst$hiLo, preds=if_else(afin_rf_tst_
preds[,2]>bThr,1,-1)))
```

```
## Confusion Matrix and Statistics
##
##       preds
## actual    -1     1
##     -1  3567  1039
##     1   1493 11901
##
##                Accuracy : 0.8593
##                  95% CI : (0.8542, 0.8644)
##     No Information Rate : 0.7189
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6422
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7049
##             Specificity : 0.9197
##          Pos Pred Value : 0.7744
##          Neg Pred Value : 0.8885
##              Prevalence : 0.2811
##          Detection Rate : 0.1982
##    Detection Prevalence : 0.2559
##       Balanced Accuracy : 0.8123
##
##        'Positive' Class : -1
##
```

Naive Bayes for AFINN Dictionary

```
nb_afin <- naiveBayes(hiLo ~ ., data=rev_senti_afin_dat %>% select(-review_id))
rev_senti_afin_NBTrn <- predict(nb_nrc, rev_senti_afin_dat_trn, type = "raw")
rev_senti_afin_NBTst <- predict(nb_nrc, rev_senti_afin_dat_tst, type = "raw")
```

```
table(actual= rev_senti_afin_dat_trn$hiLo, predicted= rev_senti_afin_NBTrn[,2]>0.5)
```

```
##       predicted
## actual FALSE  TRUE
##     -1  2239  2354
##     1   1975 11432
```

```
table(actual= rev_senti_afin_dat_tst$hiLo, predicted= rev_senti_afin_NBTst[,2]>0.5)
```

```
##       predicted
## actual FALSE  TRUE
##     -1  2252  2354
##     1   1976 11418
```

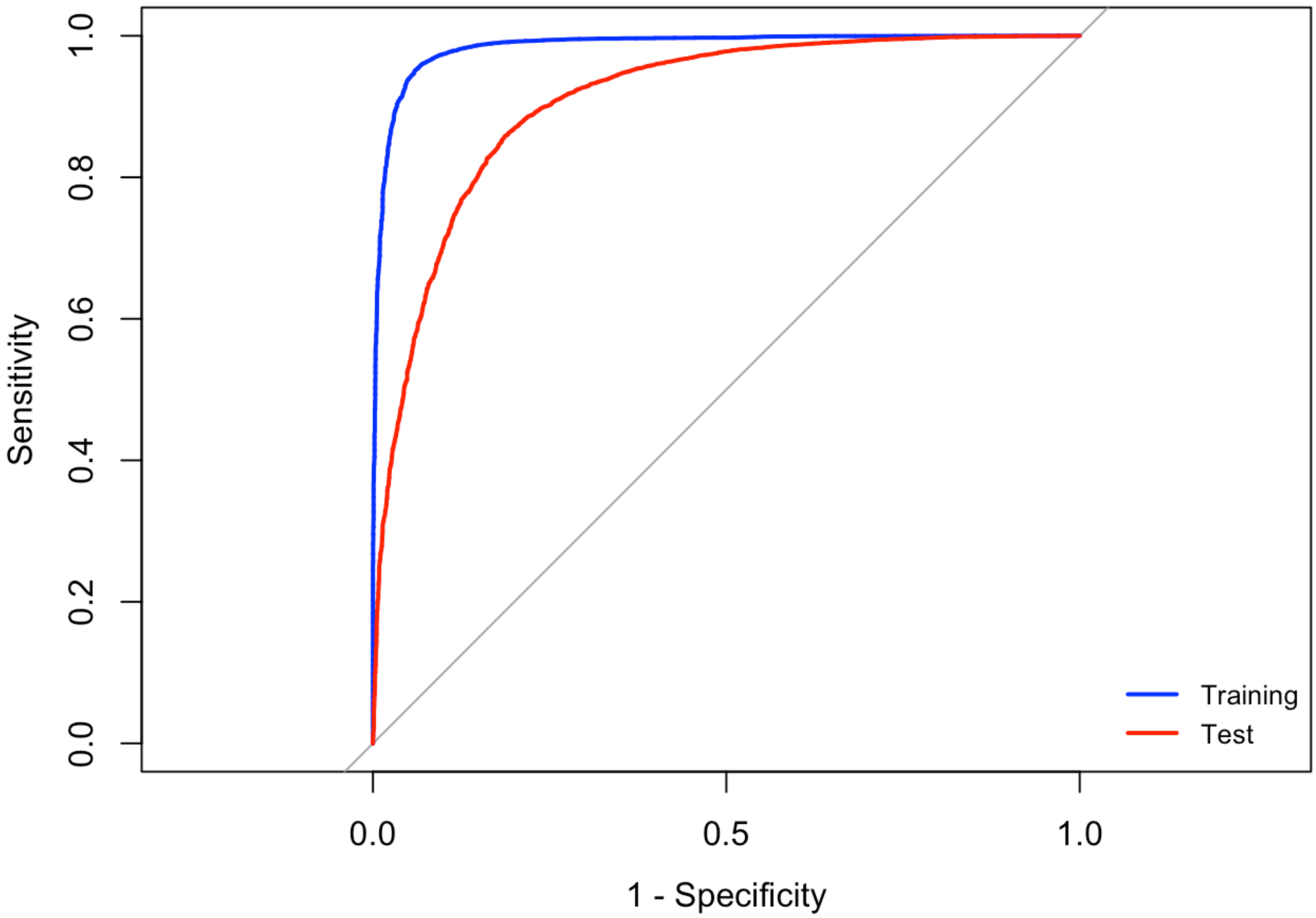
```
auc(as.numeric(rev_senti_afin_dat_trn$hiLo), rev_senti_afin_NBTrn[,2])
```

```
## Area under the curve: 0.7422
```

```
auc(as.numeric(rev_senti_afin_dat_tst$hiLo), rev_senti_afin_NBTst[,2])
```

```
## Area under the curve: 0.7469
```

```
rocTrn_nb = roc(rev_senti_afin_dat_trn$hiLo, rev_senti_afin_NBTrn[,2], levels=c(-1, 1
))
rocTst_nb = roc(rev_senti_afin_dat_tst$hiLo, rev_senti_afin_NBTst[,2], levels=c(-1, 1
))
plot.roc(rocTrn, col='blue', legacy.axes = TRUE)
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),
        col=c("blue", "red"), lwd=2, cex=0.8, bty='n')
```

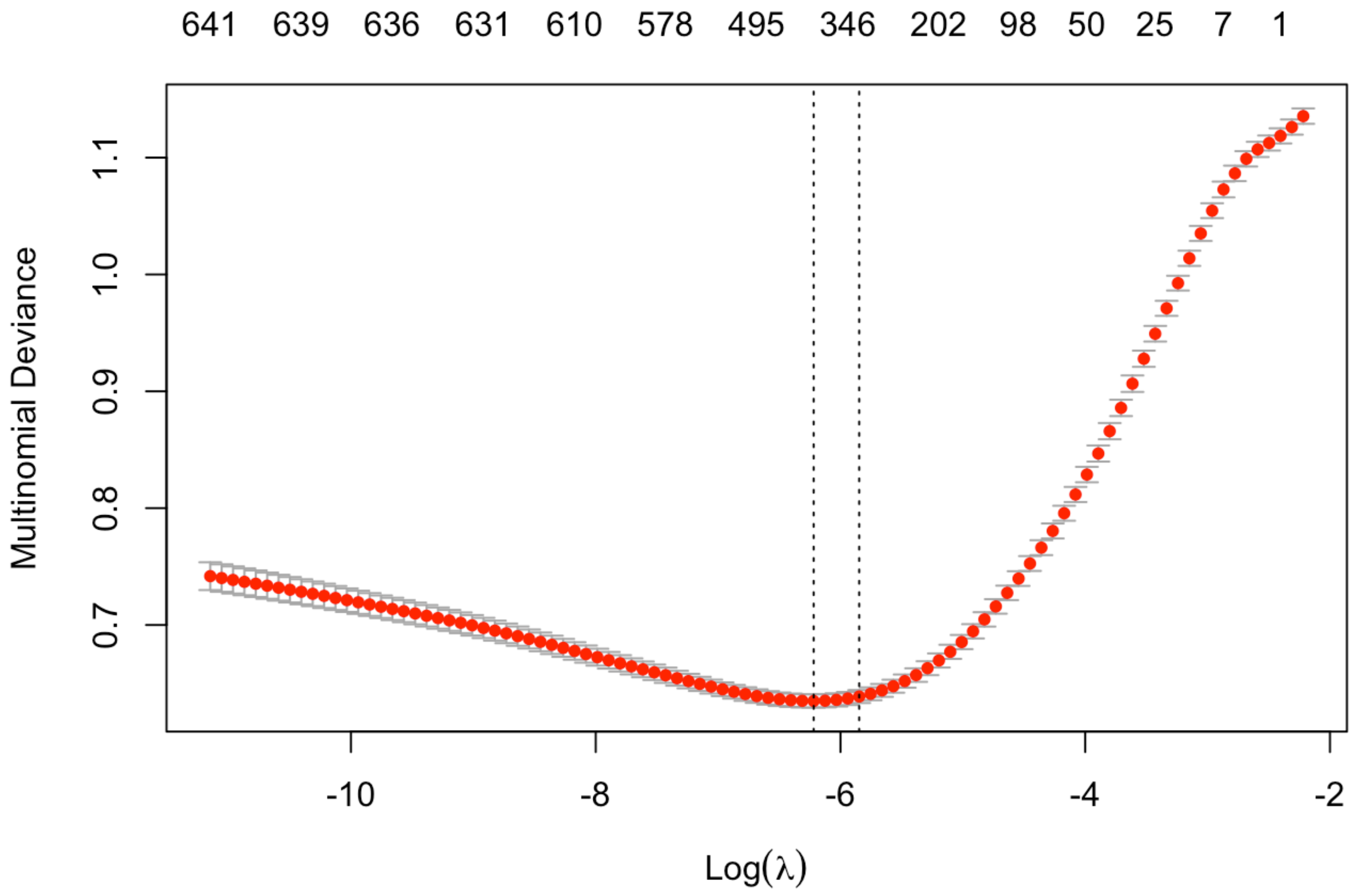



```
confusion_matrix_afin_nb = table(actual=rev_senti_afin_dat_trn$hiLo, predicted=rev_se
nti_afin_dat_tst$hiLo )
confusionMatrix(confusion_matrix_afin_nb)
```

```
## Confusion Matrix and Statistics
##
##       predicted
## actual   -1    1
##     -1 1173 3420
##     1  3433 9974
##
##                Accuracy : 0.6193
##                  95% CI : (0.6121, 0.6264)
##     No Information Rate : 0.7441
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : -7e-04
##
##  Mcnemar's Test P-Value : 0.8847
##
##             Sensitivity : 0.25467
##             Specificity : 0.74466
##          Pos Pred Value : 0.25539
##          Neg Pred Value : 0.74394
##              Prevalence : 0.25589
##          Detection Rate : 0.06517
##    Detection Prevalence : 0.25517
##       Balanced Accuracy : 0.49966
##
##        'Positive' Class : -1
##
```

Logistic Regression for AFINN Dictionary

```
library(glmnet)
Log_Reg_afin_x <- rev_senti_afin_dat_trn %>% select(-hiLo,-review_id)
Log_Reg_afin_y <- rev_senti_afin_dat_trn$hiLo

Log_Reg_afin_x_Tst <- rev_senti_afin_dat_tst %>% select(-hiLo,,-review_id)
Log_Reg_afin_y_Tst <- rev_senti_afin_dat_tst$hiLo


set.seed(231)
cvglmnet_com <- cv.glmnet(data.matrix(Log_Reg_afin_x), Log_Reg_afin_y,
                        family = "multinomial",
                        nfolds = 5,
                        alpha = 1)
plot(cvglmnet_com)
```




```
########## variable importance glmnet
library(vip)
tb1 <- vi_model(cvglmnet_com)
arrange(tb1,desc(Importance),Variable)
```

```
## # A tibble: 644 × 3
##    Variable   Importance Sign
##    <chr>           <dbl> <chr>
##  1 unclear          9.20 POS
##  2 apology          7.45 POS
##  3 excellent        7.36 NEG
##  4 amaze            7.12 NEG
##  5 perfect          6.93 NEG
##  6 terrible         6.52 POS
##  7 favorite         6.49 NEG
##  8 love             6.27 NEG
##  9 gross            6.18 POS
## 10 poor             5.97 POS
## # … with 634 more rows
```

```r
sort(tb1$Importance, decreasing = TRUE) %>% view()

library(caret)
#confusion_matrix for Trn
glm_com_train_pred <- predict(cvglmnet_com, data.matrix(Log_Reg_afin_x),s=cvglmnet_co
m$lambda.1se,type="class")
glm_com_train_pred <- factor(glm_com_train_pred, levels=c(1,-1))
Log_Reg_afin_y_Tst2 <- factor(Log_Reg_afin_y, levels=c(1,-1))
confusionMatrix (glm_com_train_pred,Log_Reg_afin_y_Tst2, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1    -1
##         1  12954  1760
##         -1   453  2833
##
##                Accuracy : 0.8771
##                  95% CI : (0.8722, 0.8818)
##     No Information Rate : 0.7448
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6432
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9662
##             Specificity : 0.6168
##          Pos Pred Value : 0.8804
##          Neg Pred Value : 0.8621
##              Prevalence : 0.7448
##          Detection Rate : 0.7197
##    Detection Prevalence : 0.8174
##       Balanced Accuracy : 0.7915
##
##        'Positive' Class : 1
##
```

```r
#confusion_matrix for Tst
glm_com_test_pred <- predict(cvglmnet_com, data.matrix(Log_Reg_afin_x_Tst),s=cvglmnet
_com$lambda.1se,type="class")
glm_com_test_pred <- factor(glm_com_test_pred, levels=c(1,-1))
Log_Reg_afin_y_Tst2 <- factor(Log_Reg_afin_y_Tst, levels=c(1,-1))
confusionMatrix (glm_com_test_pred,Log_Reg_afin_y_Tst2, positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     1    -1
##         1  12886  1827
##         -1   508  2779
##
##                Accuracy : 0.8703
##                  95% CI : (0.8653, 0.8752)
##     No Information Rate : 0.7441
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.624
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9621
##             Specificity : 0.6033
##          Pos Pred Value : 0.8758
##          Neg Pred Value : 0.8455
##              Prevalence : 0.7441
##          Detection Rate : 0.7159
##    Detection Prevalence : 0.8174
##       Balanced Accuracy : 0.7827
##
##        'Positive' Class : 1
##
```

Combined Dictionaries

Lets merge all the matched words from all three dictionaries into a single combined dictionary. The single combined dictionary matched data set consists of words where each word can have several sentiments from all the dictionaries.

```
#Combined Dict.
names(from_afin_dict)[names(from_afin_dict) == "value"] = "sentiment"
#Dimensions for matched words from all three dictionaries

#Converting the sentiment variable in AFINN dictionary to character
from_afin_dict = from_afin_dict %>% mutate(sentiment = as.character(sentiment))
#combine matched words from the three dictionaries
comb_dict = rbind(from_bing_dict, from_nrc_dict, from_afin_dict)
#comb_dict %>% dim()
#Dimensions for the distinct word tokens in comb_dict
#comb_dict %>% distinct(word) %>% dim()
#remove duplicates from comb_dict
comb_dict_1 = comb_dict[,-2]
comb_dict_1 = comb_dict_1[!duplicated(comb_dict_1), ]
#Dimensions for rrSenti_combo
#comb_dict_1 %>% dim()
#Dimensions for the distinct word tokens in comb_dict_1
#comb_dict_1 %>% distinct(word) %>% dim()
#create Document Term Matrix
senti_comb_data = comb_dict_1 %>%  pivot_wider(id_cols = c(review_id,starsReview), na
mes_from = word, values_from = tf_idf)  %>% ungroup()
#filter out the reviews with starsReview=3
#calculate hiLo sentiment(1 is assigned to 4 and 5/-1 is assigned to 1 and 2)
senti_comb_data = senti_comb_data %>% filter(starsReview!=3) %>% mutate(hiLo=ifelse(s
tarsReview<=2, -1, 1)) %>% select(-starsReview)
#replace all NAs with zero
senti_comb_data = senti_comb_data %>% replace(., is.na(.), 0)
#convert hiLo from num to factor
senti_comb_data$hiLo = as.factor(senti_comb_data$hiLo)
#no of reviews with 1, -1 class
senti_comb_data %>% group_by(hiLo) %>% tally()
```

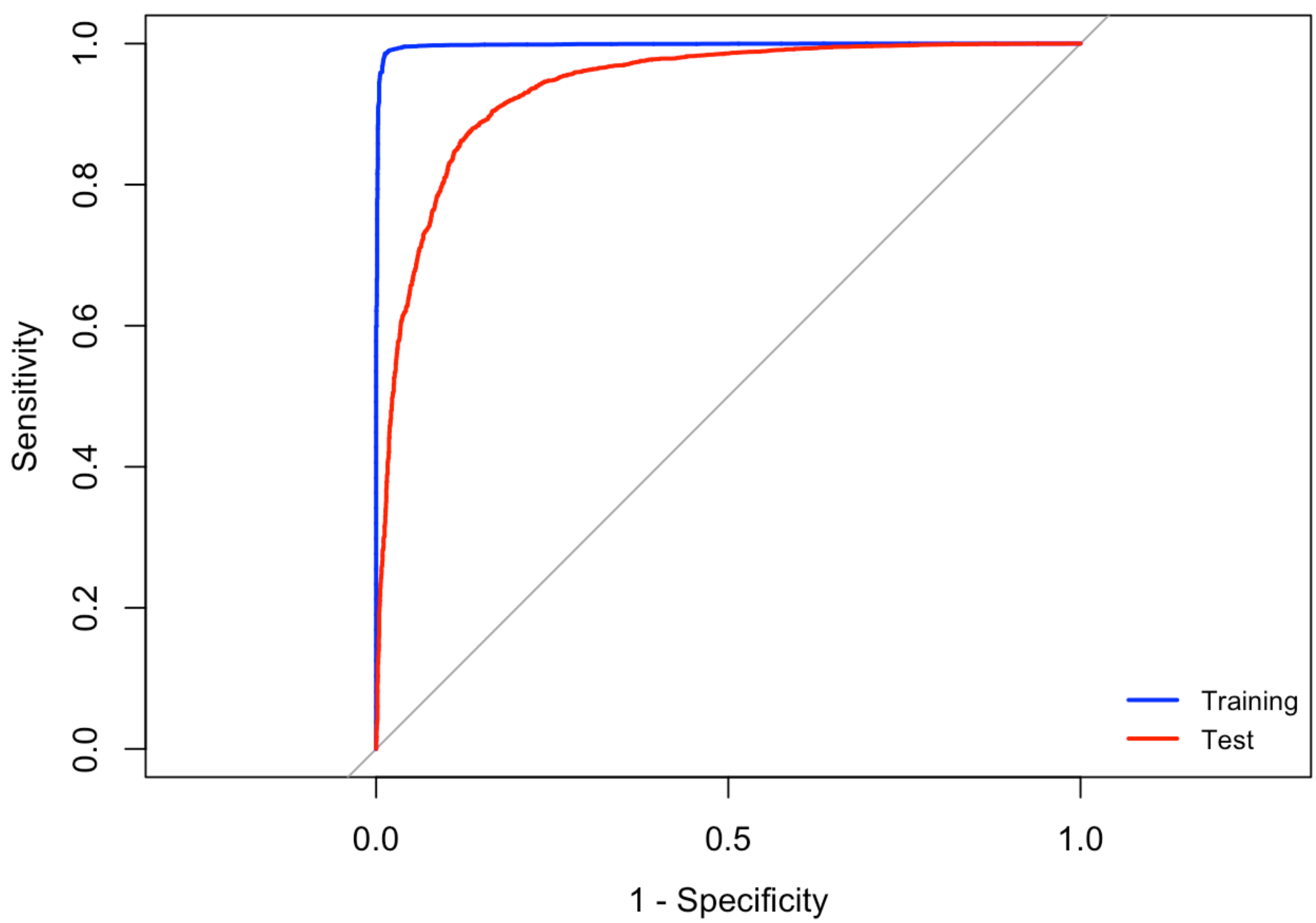
```
## # A tibble: 2 × 2
##   hiLo      n
##   <fct> <int>
## 1 -1     9906
## 2 1     28563
```

```
set.seed(213)
senti_comb_data_16k = senti_comb_data[sample(nrow(senti_comb_data),16000),]
senti_comb_data_16k_split = initial_split(senti_comb_data_16k, 0.5)
senti_comb_data_16k_trn = training(senti_comb_data_16k_split)
senti_comb_data_16k_tst  = testing(senti_comb_data_16k_split)
```

```
rfModel_comb = ranger(dependent.variable.name = "hiLo", data=senti_comb_data_16k_trn
%>% select(-review_id), num.trees = 200, importance='permutation', probability = TRUE
)
```

```
## Computing permutation importance.. Progress: 93%. Estimated remaining time: 2 seco
nds.
```

```
#Obtain predictions, and calculate performance
comb_rf_trn_preds = predict(rfModel_comb, senti_comb_data_16k_trn %>% select(-review_
id))$predictions
comb_rf_tst_preds = predict(rfModel_comb, senti_comb_data_16k_tst %>% select(-review_
id))$predictions
#The optimal threshold from the ROC analyses
library(pROC)
rocTrn = roc(senti_comb_data_16k_trn$hiLo, comb_rf_trn_preds[,2], levels=c(-1, 1))
rocTst = roc(senti_comb_data_16k_tst$hiLo, comb_rf_tst_preds[,2], levels=c(-1, 1))
plot.roc(rocTrn, col='blue', legacy.axes = TRUE)
plot.roc(rocTst, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),col=c("blue", "red"), lwd=2, cex=0
.8, bty='n')
```




```
#Best threshold from ROC analyses
bThr = coords(rocTrn, "best", ret="threshold", transpose = FALSE)
bThr = as.numeric(bThr)
bThr
```

```
## [1] 0.6131181
```

```
#Confusion Matrix at bThr for Trn and Tst dataset
confusionMatrix(table(actual=senti_comb_data_16k_trn$hiLo, preds=if_else(comb_rf_trn_
preds[,2]>bThr,1,-1)))
```

```
## Confusion Matrix and Statistics
##
##       preds
## actual   -1    1
##     -1 2065   26
##     1    83 5826
##
##                Accuracy : 0.9864
##                  95% CI : (0.9836, 0.9888)
##     No Information Rate : 0.7315
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.965
##
##  Mcnemar's Test P-Value : 8.148e-08
##
##             Sensitivity : 0.9614
##             Specificity : 0.9956
##          Pos Pred Value : 0.9876
##          Neg Pred Value : 0.9860
##              Prevalence : 0.2685
##          Detection Rate : 0.2581
##    Detection Prevalence : 0.2614
##       Balanced Accuracy : 0.9785
##
##        'Positive' Class : -1
##
```

```
confusionMatrix(table(actual=senti_comb_data_16k_tst$hiLo, preds=if_else(comb_rf_tst_
preds[,2]>bThr,1,-1)))
```

```
## Confusion Matrix and Statistics
##
##       preds
## actual   -1    1
##     -1 1645  435
##     1   425 5495
##
##                Accuracy : 0.8925
##                  95% CI : (0.8855, 0.8992)
##     No Information Rate : 0.7412
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7202
##
##  Mcnemar's Test P-Value : 0.7589
##
##             Sensitivity : 0.7947
##             Specificity : 0.9266
##          Pos Pred Value : 0.7909
##          Neg Pred Value : 0.9282
##              Prevalence : 0.2587
##          Detection Rate : 0.2056
##    Detection Prevalence : 0.2600
##       Balanced Accuracy : 0.8607
##
##        'Positive' Class : -1
##
```

```
col1 = c('BING','NRC','AFINN','Combined')
col2 = c(96.52,97.57,94.84,98.65)
col3 = c(88.54,87.12,85.93,89.56)
colna = c('Dictionary','Accuracy (%) on Training Data','Accuracy (%) on Test Data')
metric_df = data.frame(cbind(col1,col2,col3))
names(metric_df) = colna
metric_df
```

```
##   Dictionary Accuracy (%) on Training Data Accuracy (%) on Test Data
## 1       BING                         96.52                     88.54
## 2        NRC                         97.57                     87.12
## 3      AFINN                         94.84                     85.93
## 4   Combined                         98.65                     89.56
```

Question 6

Consider some of the attributes for restaurants – this is specified as a list of values for various attributes in the ‘attributes’ column. Extract different attributes (see note below).

a. Consider a few interesting attributes and summarize how many restaurants there are by values of these attributes; examine if star ratings vary by these attributes.

b. For one of your models (choose your ‘best’ model from above), does prediction accuracy vary by certain restaurant attributes? You do not need to look into all attributes; choose a few which you think may be interesting, and examine these.

```
x<- df %>% select (review_id, attributes)
paste(x[1,2])
```

```
## [1] "Alcohol: none|Ambience: {'romantic': False, 'intimate': False, 'classy': Fals
e, 'hipster': False, 'divey': False, 'touristy': False, 'trendy': False, 'upscale': F
alse, 'casual': True}|BikeParking: True|BusinessAcceptsCreditCards: True|BusinessPark
ing: {'garage': False, 'street': False, 'validated': False, 'lot': True, 'valet': Fal
se}|Caters: False|DriveThru: False|GoodForKids: True|HasTV: True|NoiseLevel: average|
OutdoorSeating: False|RestaurantsAttire: casual|RestaurantsDelivery: False|Restaurant
sGoodForGroups: True|RestaurantsPriceRange2: 2|RestaurantsReservations: False|Restaur
antsTableService: True|RestaurantsTakeOut: True|WheelchairAccessible: True|WiFi: no|G
oodForMeal: {'dessert': False, 'latenight': False, 'lunch': True, 'dinner': True, 'br
eakfast': False, 'brunch': False}"
```

```
x2<-x %>% mutate (atts=str_split( attributes,'\\|')) %>% unnest(atts)

x3<-x2 %>% cbind (str_split_fixed ( x2$atts, ":", 2))

colnames(x3)[4] <- 'attName'
colnames(x3)[5] <- 'attValue'
colnames(x3)
```

```
## [1] "review_id"  "attributes" "atts"       "attName"    "attValue"
```

```
x3<-x3 %>% select (-c(attributes ,atts))
x3<-x3 %>% filter(str_length(x3$attName) > 0)

x4<-x3 %>% pivot_wider(names_from = attName, values_from = attValue)

dim(x4)
```

```
## [1] 44884    38
```

```
glimpse(x4)
```

```
## Rows: 44,884
## Columns: 38
## $ review_id                  <chr> "K24CBfrL8nQXlGOmFInmVw", "245Q0ZiJESuxuPzJ…
## $ Alcohol                    <chr> " none", " beer_and_wine", " full_bar", " n…
## $ Ambience                   <chr> " {'romantic': False, 'intimate': False, 'c…
## $ BikeParking                <chr> " True", " True", " True", " True", " True"…
## $ BusinessAcceptsCreditCards <chr> " True", " True", " True", " True", " True"…
## $ BusinessParking            <chr> " {'garage': False, 'street': False, 'valid…
## $ Caters                     <chr> " False", " True", " False", " True", " Tru…
## $ DriveThru                  <chr> " False", NA, NA, NA, NA, NA, NA, NA, NA, N…
## $ GoodForKids                <chr> " True", " True", " False", " True", " True…
## $ HasTV                      <chr> " True", " True", " False", " False", " Fal…
## $ NoiseLevel                 <chr> " average", " average", " average", " avera…
## $ OutdoorSeating             <chr> " False", " False", " False", " False", " F…
## $ RestaurantsAttire          <chr> " casual", " casual", " dressy", " casual",…
## $ RestaurantsDelivery        <chr> " False", " False", " False", " True", " Tr…
## $ RestaurantsGoodForGroups   <chr> " True", " True", " True", " True", " True"…
## $ RestaurantsPriceRange2     <chr> " 2", " 2", " 4", " 2", " 2", " 2", " 2", "…
## $ RestaurantsReservations    <chr> " False", " False", " True", " False", " Tr…
## $ RestaurantsTableService    <chr> " True", " True", " True", " False", " True…
## $ RestaurantsTakeOut         <chr> " True", " True", " False", " True", " True…
## $ WheelchairAccessible       <chr> " True", NA, " True", " True", " True", " T…
## $ WiFi                       <chr> " no", " no", " no", " no", " no", " free",…
## $ GoodForMeal                <chr> " {'dessert': False, 'latenight': False, 'l…
## $ GoodForDancing             <chr> NA, NA, " False", NA, NA, NA, NA, NA, NA, "…
## $ HappyHour                  <chr> NA, NA, " True", NA, NA, NA, NA, NA, NA, " …
## $ BusinessAcceptsBitcoin     <chr> NA, NA, NA, NA, NA, " False", NA, NA, NA, N…
## $ BYOB                       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, " False…
## $ BYOBCorkage                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, " yes_f…
## $ BestNights                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, " {'mon…
## $ CoatCheck                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, " False…
## $ Corkage                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, " False…
## $ Music                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, " {'dj'…
## $ Smoking                    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, " no", …
## $ DogsAllowed                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ Open24Hours                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ ByAppointmentOnly          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ RestaurantsCounterService  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ AgesAllowed                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
## $ DietaryRestrictions        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,…
```

```
#Now we analyze 'Ambience'

paste(x4[1,3])
```

```
## [1] " {'romantic': False, 'intimate': False, 'classy': False, 'hipster': False, 'd
ivey': False, 'touristy': False, 'trendy': False, 'upscale': False, 'casual': True}"
```

```
x5<-x4 %>% mutate (amb = str_split(Ambience, ","))

dim(x4)
```

```
## [1] 44884    38
```

```
dim(x5)
```

```
## [1] 44884    39
```

```
typeof(x5$amb)
```

```
## [1] "list"
```

```
x5$amb[1]
```

```
## [[1]]
## [1] " {'romantic': False" " 'intimate': False"  " 'classy': False"
## [4] " 'hipster': False"   " 'divey': False"     " 'touristy': False"
## [7] " 'trendy': False"    " 'upscale': False"   " 'casual': True}"
```

```
x5$amb[1000]
```

```
## [[1]]
## [1] " {'romantic': False" " 'intimate': False"  " 'classy': False"
## [4] " 'hipster': False"   " 'divey': False"     " 'touristy': False"
## [7] " 'trendy': False"    " 'upscale': False"   " 'casual': True}"
```

```r
#creating the function

extractAmbience<-function(q)
{  sub(":.*","", q[which(str_extract(q,"True") == "True")])
}

x6<-x5 %>% mutate (amb = lapply (amb,extractAmbience ) )

#how many examples by different values for 'Ambience'

x6 %>% group_by(amb) %>% tally() %>% view()
y <- df %>% select(review_id, starsReview)



x7 <- merge(x6,y)
x7 %>% filter(str_detect (amb,'casual')) %>% summarise(n(),AvgStar = mean(starsReview
))
```

```
##     n()  AvgStar
## 1 33031 3.672792
```

```r
x7 %>% filter(str_detect (amb,'classy')) %>% summarise(n(),AvgStar = mean(starsReview
))
```

```
##    n()  AvgStar
## 1 1909 3.861184
```

```r
#Now we analyze 'GoodForMeal'

paste(x4[1,7])
```

```
## [1] " False"
```

```r
x5<-x4 %>% mutate (GdFrMl = str_split (GoodForMeal, ","))

dim(x4)
```

```
## [1] 44884    38
```

```r
dim(x5)
```

```
## [1] 44884    39
```

```
typeof(x5$GdFrMl)
```

```
## [1] "list"
```

```
x5$GdFrMl[1]
```

```
## [[1]]
## [1] " {'dessert': False"  " 'latenight': False" " 'lunch': True"
## [4] " 'dinner': True"     " 'breakfast': False" " 'brunch': False}"
```

```
x5$GdFrMl[1000]
```

```
## [[1]]
## [1] " {'dessert': False"  " 'latenight': False" " 'lunch': False"
## [4] " 'dinner': False"    " 'breakfast': True"  " 'brunch': True}"
```

```
#creating the function

extractgood4meal<-function(q)
{  sub(":.*","", q[which(str_extract(q,"True") == "True")])
}

x6<-x5 %>% mutate (GdFrMl = lapply (GdFrMl, extractgood4meal ) )

#how many examples by different values for 'Good For Meal'

x6%>%group_by(GdFrMl) %>% tally() %>% view()

x7 <- merge(x6,y)
x7%>%filter(str_detect (GdFrMl,'lunch'))  %>% summarise(n(),AvgStar = mean(starsRevie
w))
```

```
##     n()  AvgStar
## 1 28991 3.672657
```

```
x7%>%filter(str_detect (GdFrMl,'dinner')) %>% summarise(n(),AvgStar = mean(starsRevie
w))
```

```
##     n()  AvgStar
## 1 26777 3.681816
```

```r
#Now we analyze 'BusinessParking'

paste(x4[1,5])
```

```
## [1] " True"
```

```r
x5 <- x4 %>% mutate( bsnsPrk = str_split( BusinessParking, ","))

dim(x4)
```

```
## [1] 44884    38
```

```r
dim(x5)
```

```
## [1] 44884    39
```

```r
typeof(x5$bsnsPrk)
```

```
## [1] "list"
```

```r
x5$bsnsPrk[1]
```

```
## [[1]]
## [1] " {'garage': False"   " 'street': False"    " 'validated': False"
## [4] " 'lot': True"        " 'valet': False}"
```

```r
x5$bsnsPrk[1000]
```

```
## [[1]]
## [1] " {'garage': False"   " 'street': True"     " 'validated': False"
## [4] " 'lot': True"        " 'valet': False}"
```

```
#creating the function

extractBuspark<-function(q)
{  sub(":.*","", q[which(str_extract(q, "True") == "True")])
}

x6<-x5%>% mutate (bsnsPrk=lapply(bsnsPrk, extractBuspark ) )

#how many examples by different values for 'Bus Park'

x6%>% group_by(bsnsPrk) %>% tally() %>% view()

x7 <- merge(x6,y)
x7%>% filter(str_detect (bsnsPrk,'lot'))%>% summarise(n(),AvgStar = mean(starsReview)
)
```

```
##     n()  AvgStar
## 1 29168 3.683729
```

```
x7%>% filter(str_detect (bsnsPrk,'street'))%>% summarise(n(),AvgStar = mean(starsRevi
ew))
```

```
##    n()  AvgStar
## 1 9519 3.833701
```

```
####################################

x7 = x7 %>% mutate(hiLo=ifelse(starsReview<=2,-1, ifelse(starsReview>=4, 1, 0 )))

str(x7)
```

```
## 'data.frame':    44884 obs. of  41 variables:
##  $ review_id                 : chr  "__0PpIOWdiB5VG5NiHvQtQ" "__14t3nE7EVUVq3tnC5A
qQ" "__2BD1YbmOQCERuqAUkSyg" "__6ze-c46vN0_edDEdyB8w" ...
##  $ Alcohol                   : chr  " none" " none" " none" " full_bar" ...
##  $ Ambience                  : chr  " {'romantic': False, 'intimate': False, 'clas
sy': False, 'hipster': False, 'divey': False, 'touristy': False, '"| __truncated__ "
{'romantic': False, 'intimate': False, 'classy': False, 'hipster': False, 'divey': Fa
lse, 'touristy': False, '"| __truncated__ " {'romantic': False, 'intimate': False, 'c
lassy': False, 'hipster': False, 'divey': False, 'touristy': False, '"| __truncated__
" {'romantic': False, 'intimate': False, 'classy': False, 'hipster': False, 'divey':
False, 'touristy': False, '"| __truncated__ ...
##  $ BikeParking               : chr  " True" " True" " True" " True" ...
##  $ BusinessAcceptsCreditCards: chr  " True" " True" " False" " True" ...
##  $ BusinessParking           : chr  " {'garage': False, 'street': False, 'validate
d': False, 'lot': True, 'valet': False}" " {'garage': False, 'street': False, 'valida
```

```
ted': False, 'lot': True, 'valet': False}" " {'garage': False, 'street': True, 'valid
ated': False, 'lot': False, 'valet': False}" " {'garage': False, 'street': False, 'va
lidated': False, 'lot': True, 'valet': False}" ...
##  $ Caters                    : chr  " True" " True" " False" " True" ...
##  $ DriveThru                 : chr  NA NA NA NA ...
##  $ GoodForKids               : chr  " True" " True" " True" " True" ...
##  $ HasTV                     : chr  " True" " False" " True" " True" ...
##  $ NoiseLevel                : chr  " average" " average" " average" " average" ..
.
##  $ OutdoorSeating            : chr  " True" " False" " False" " True" ...
##  $ RestaurantsAttire         : chr  " casual" " casual" " casual" " casual" ...
##  $ RestaurantsDelivery       : chr  " False" " True" " False" " False" ...
##  $ RestaurantsGoodForGroups  : chr  " True" " True" " True" " True" ...
##  $ RestaurantsPriceRange2    : chr  " 1" " 1" " 1" " 2" ...
##  $ RestaurantsReservations   : chr  " True" " False" " False" " True" ...
##  $ RestaurantsTableService   : chr  " True" " False" " False" " True" ...
##  $ RestaurantsTakeOut        : chr  " True" " True" " True" " True" ...
##  $ WheelchairAccessible      : chr  " True" " False" " True" " True" ...
##  $ WiFi                      : chr  " free" " no" " no" " no" ...
##  $ GoodForMeal               : chr  " {'dessert': False, 'latenight': False, 'lunc
h': False, 'dinner': False, 'breakfast': True, 'brunch': True}" " {'dessert': False,
'latenight': False, 'lunch': False, 'dinner': True, 'breakfast': False, 'brunch': Fal
se}" " {'dessert': False, 'latenight': False, 'lunch': False, 'dinner': True, 'breakf
ast': False, 'brunch': False}" " {'dessert': False, 'latenight': False, 'lunch': True
, 'dinner': True, 'breakfast': False, 'brunch': False}" ...
##  $ GoodForDancing            : chr  NA NA NA NA ...
##  $ HappyHour                 : chr  NA NA NA NA ...
##  $ BusinessAcceptsBitcoin    : chr  " False" NA NA NA ...
##  $ BYOB                      : chr  NA NA NA NA ...
##  $ BYOBCorkage               : chr  NA NA NA NA ...
##  $ BestNights                : chr  NA NA NA NA ...
##  $ CoatCheck                 : chr  NA NA NA NA ...
##  $ Corkage                   : chr  NA NA NA NA ...
##  $ Music                     : chr  NA NA NA NA ...
##  $ Smoking                   : chr  NA NA NA NA ...
##  $ DogsAllowed               : chr  " True" NA NA NA ...
##  $ Open24Hours               : chr  NA NA NA NA ...
##  $ ByAppointmentOnly         : chr  NA NA NA NA ...
##  $ RestaurantsCounterService : chr  NA NA " True" NA ...
##  $ AgesAllowed               : chr  NA NA NA NA ...
##  $ DietaryRestrictions       : chr  NA NA NA NA ...
##  $ bsnsPrk                   :List of 44884
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " {'garage'"
##   ..$ : chr  " 'street'" " 'lot'"
##   ..$ : chr " 'street'"
```

```
##   ..$ : chr " 'lot'"
##   ..$ : chr
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr
##   ..$ : chr " 'lot'"
##   ..$ : chr
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " {'garage'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr  " {'garage'" " 'lot'" " 'valet'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr  " 'street'" " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr  " 'street'" " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr  " {'garage'" " 'lot'" " 'valet'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr
##   ..$ : chr  " {'garage'" " 'lot'" " 'valet'"
##   ..$ : chr " {'garage'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr  " {'garage'" " 'validated'"
##   ..$ : chr " 'lot'"
##   ..$ : chr  " 'street'" " 'lot'"
##   ..$ : chr
```

```
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr  " 'street'" " 'lot'"
##   ..$ : chr  " 'street'" " 'lot'"
##   ..$ : chr
##   ..$ : chr " 'lot'"
##   ..$ : chr " {'garage'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'valet'"
##   ..$ : chr " {'garage'"
##   ..$ : chr
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr
##   ..$ : chr " 'lot'"
##   ..$ : chr
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr
##   ..$ : chr  " 'street'" " 'lot'"
##   ..$ : chr " 'street'"
##   ..$ : chr " {'garage'"
##   ..$ : chr " {'garage'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'lot'"
##   ..$ : chr " 'valet'"
##   .. [list output truncated]
##  $ starsReview              : int  5 5 5 5 5 1 3 3 1 5 ...
##  $ hiLo                     : num  1 1 1 1 1 -1 0 0 -1 1 ...
```

```
x8 = subset(x7, select = -c(bsnsPrk) )

revAttDTM <- x8
dim(revAttDTM)
```

```
## [1] 44884    40
```

```
revAttDTM<-revAttDTM %>% replace(., is.na(.), 0)
revAttDTM$hiLo<-as.factor(revAttDTM$hiLo)
revAttDTM_split<- initial_split(revAttDTM, 0.5)
revAttDTM_trn<- training(revAttDTM_split)
revAttDTM_tst<- testing(revAttDTM_split)

rfModel_Att<-ranger(hiLo ~., data=revAttDTM_trn %>% select(-review_id), num.trees = 5
00, importance='permutation', probability = TRUE)

importance(rfModel_Att)
```

```
##                    Alcohol                   Ambience
##               4.854345e-03               2.716330e-03
##                BikeParking BusinessAcceptsCreditCards
##               1.613078e-03               1.962027e-05
##            BusinessParking                     Caters
##               3.569926e-03               2.307475e-03
##                  DriveThru                 GoodForKids
##               9.560916e-04               1.322005e-03
##                      HasTV                 NoiseLevel
##               2.351322e-03               1.923918e-03
##             OutdoorSeating          RestaurantsAttire
##               2.098972e-03               3.858480e-04
##        RestaurantsDelivery   RestaurantsGoodForGroups
##               1.085008e-03               6.551073e-04
##      RestaurantsPriceRange2    RestaurantsReservations
##               2.494649e-03               2.914390e-03
##     RestaurantsTableService         RestaurantsTakeOut
##               1.881210e-03               9.809706e-04
##       WheelchairAccessible                       WiFi
##               1.757480e-03               1.653447e-03
##                GoodForMeal             GoodForDancing
##               3.718916e-03               1.552212e-03
##                  HappyHour     BusinessAcceptsBitcoin
##               1.627426e-03               1.458859e-03
##                       BYOB                BYOBCorkage
##               2.783820e-04               2.907741e-04
##                 BestNights                  CoatCheck
##               3.794522e-03               3.355488e-03
##                    Corkage                      Music
##               6.108785e-05               2.507975e-03
##                    Smoking                 DogsAllowed
##               3.645216e-03               1.930199e-03
##                Open24Hours          ByAppointmentOnly
##               3.603813e-05               2.828700e-05
##   RestaurantsCounterService                AgesAllowed
##              -5.329419e-05              -8.764349e-06
##        DietaryRestrictions                starsReview
##              -9.219963e-06               4.392072e-01
```

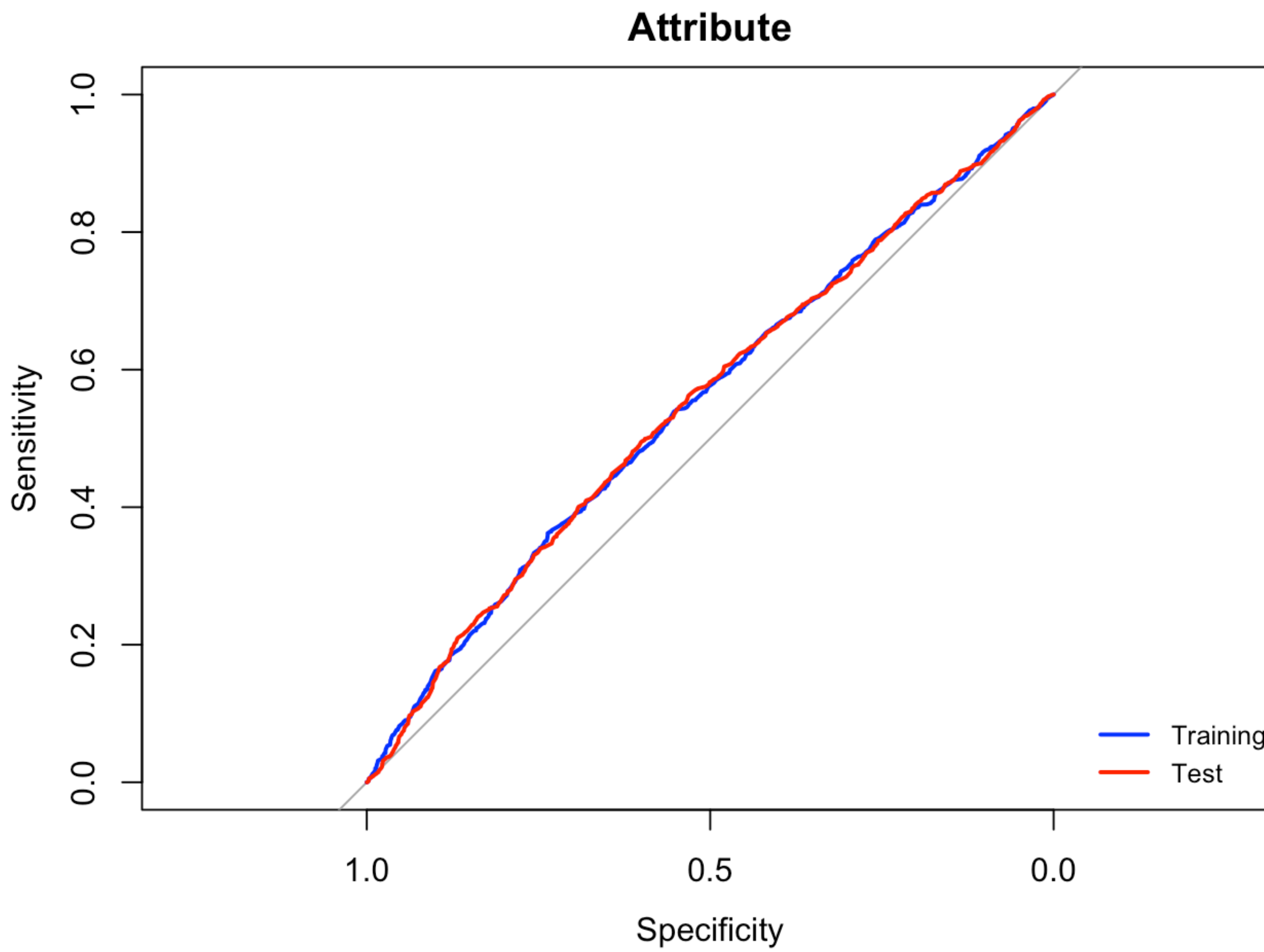
```
revAttDTM_predTrn<- predict(rfModel_Att, revAttDTM_trn %>% select(-review_id))$predic
tions
revAttDTM_predTst<- predict(rfModel_Att, revAttDTM_tst %>% select(-review_id))$predic
tions

table(actual=revAttDTM_trn$starsReview, preds=revAttDTM_predTrn[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##      1  2570    0
##      2  2316    0
##      3     7 3212
##      4  6115    0
##      5  8222    0
```

```
table(actual=revAttDTM_tst$starsReview, preds=revAttDTM_predTst[,2]>0.5)
```

```
##       preds
## actual FALSE TRUE
##      1  2700    0
##      2  2301    0
##      3     9 3165
##      4  6203    0
##      5  8064    0
```

```
rocTrn2 <- roc(revAttDTM_trn$starsReview, revAttDTM_predTrn[,2])
rocTst2 <- roc(revAttDTM_tst$starsReview, revAttDTM_predTst[,2])
plot.roc(rocTrn2, col='blue', main = "Attribute")
plot.roc(rocTst2, col='red', add=TRUE)
legend("bottomright", legend=c("Training", "Test"),col=c("blue", "red"), lwd=2, cex=0
.8, bty='n')
```



Conclusion

We can see that random forest model with combined dictionary is performing well with an accuracy of 89.56% and when we just used the dictionaries, BING dictionary performed well with accuracy being approximately 88.54%.