Data Mining for Business Project
Yelp Customer Review_Sentiment Analysis

(A) Explore the data.
From the dataset we can tell that Yelp restaurant review ratings are listed as one to five stars. The distribution shows five stars ratings with the highest number of ratings,15,084 ratings out of a total of 40,087 reviews. As to the least reviews, it is two stars with 4,094 ratings.

I consider reviews with 1 to 2 stars as negative, and 4 to 5 stars as positive, since 3 stars are the neutral reviews.

	starsReview	n
1	1	4553
2	2	4094
3	3	5561
4	4	10795
5	5	15084

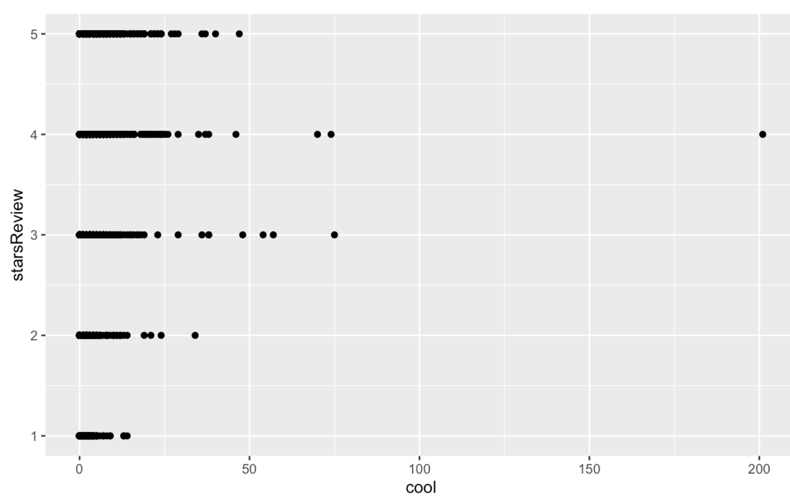
- After conducting the process of removing rare words and numbers etc, below is the most frequent and Least frequent words in the review
  xx %>% count(word, sort=TRUE) %>% view()

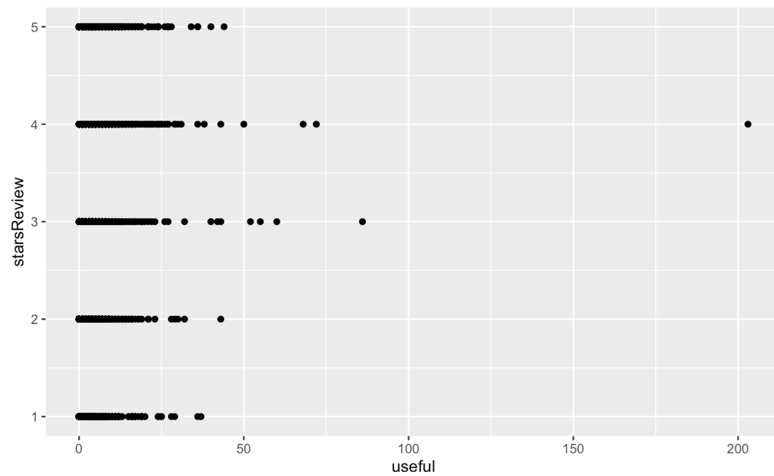
	word	n
1	food	32113
2	service	15844
3	time	12521
4	chicken	9411
5	restaurant	8833
6	nice	7907
7	menu	7574
8	love	7145
9	delicious	7090
10	bar	6202
11	friendly	6180
12	sauce	5945
13	salad	5865
14	cheese	5833
15	pizza	5829
16	lunch	5672

	word	n
1	abound	10
2	accented	10
3	accommodations	10
4	accoutrements	10
5	accused	10
6	aint	10
7	aji	10
8	aloud	10
9	ancho	10
10	angela	10
11	antipasti	10
12	anxiously	10
13	applaud	10
14	applesauce	10
15	arab	10
16	arguably	10

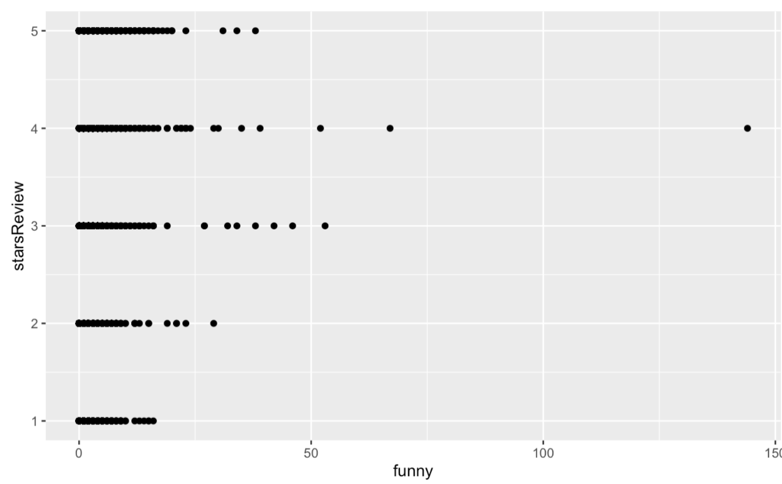
For the relationship between ‘funny’, ‘cool’, ‘useful’, I first printed out the distribution among star reviews, 1 to 5. From the following 3 plots, we can notice all 3 attributes, funny, cool, and useful, possess similar characteristics. As to the outlier in the 4-star-review, it was a popular review for “Salem's Market & Grill” which has outstanding performance on both 3 categories.

Furthermore, I looked into the relationship between 2 attributes, and they, without surprise, possess positive correlations. (Please take references from image 4 to 6)

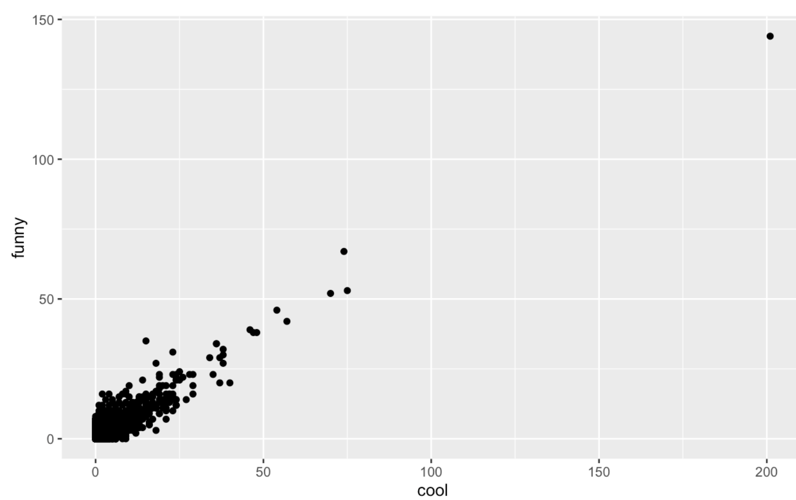




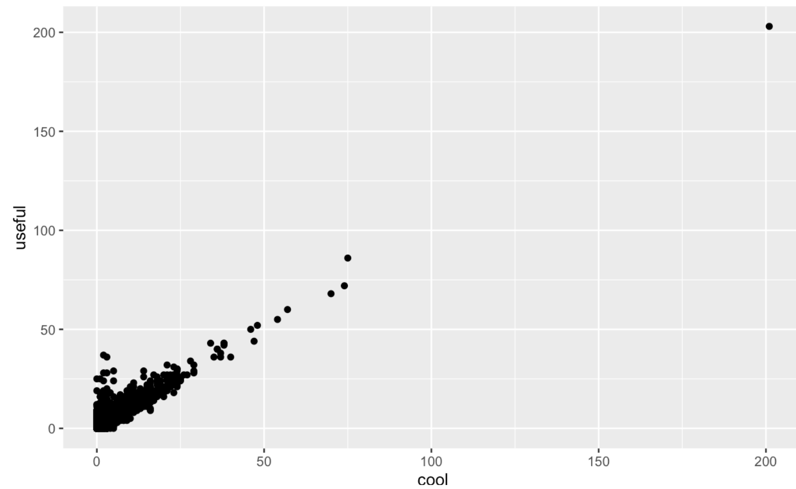
starsReview
useful

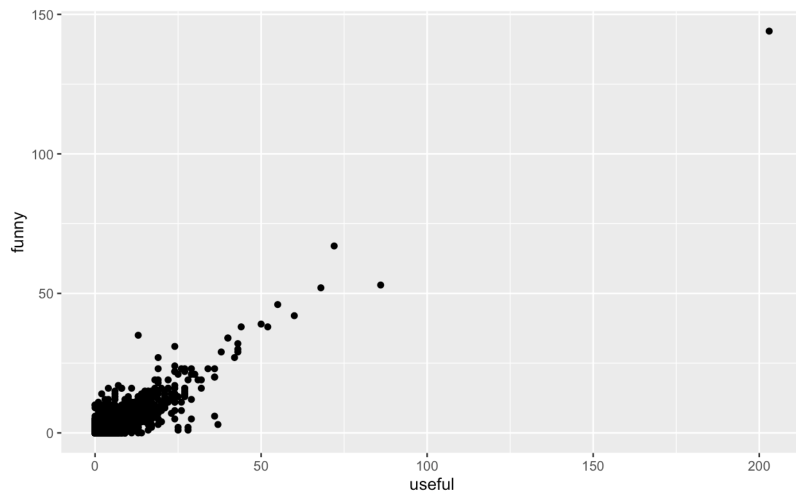


starsReview
funny



funny
cool


200
150
100
50
0
useful
0
50
100
150
200
cool


150
100
50
0
funny
0
50
100
150
200
useful

(ii)
How does star ratings for reviews relate to the star-rating given in the dataset for businesse (attribute ‘businessStars’)? (Can one be calculated from the other?)

To have a clearer view of the relationship between “starsReview” and “starsBusiness”I built a linear regression model as shown below.

starsBusiness = 3.029422 + 0.1793 * starsBusiness

```
# (a) (ii)
\#How does star ratings for reviews relate to the star-rating given in the dataset for businesse (attribute ‘businessStars’)? (Can one be calculated from the other?)
```{r pressure, echo=FALSE}
cor(resReviewsData$starsReview, resReviewsData$starsBusiness) #[1] 0.4114881
lmstars <- lm(starsBusiness ~starsReview, data =resReviewsData)
summary(lmstars)
 # we can get a statistically significant linear regression for those 2 caegories of stars review

```
```

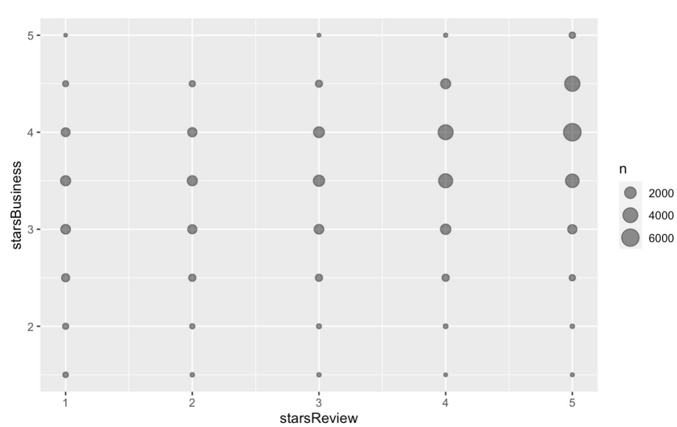
```
Call:
lm(formula = starsBusiness ~ starsReview, data = resReviewsData)

Residuals:
     Min       1Q   Median       3Q      Max
-2.42615 -0.38811  0.07385  0.43254  1.79123

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.029422   0.007808  388.00   <2e-16 ***
starsReview 0.179346   0.001984   90.39   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5405 on 40085 degrees of freedom
Multiple R-squared:  0.1693,    Adjusted R-squared:  0.1693
F-statistic:  8171 on 1 and 40085 DF,  p-value: < 2.2e-16
```

Besides the linear model, here is the plot for the relationship:
> stars_relationship <- ggplot(resReviewsData, aes(x= starsReview, y=starsBusiness)) + geom_count(alpha = 0.5) # geom_point()



After running a linear regression with the entire data, I also looked into the relationship between starsReview and starsBusiness in individual restaurants.

First, I started with finding the restaurants with top 10 reviews. And then build linear models individually.

	name	n
1	Allegro	298
2	Momo Sushi	297
3	In-N-Out Burger	294
4	Thai House	294
5	Buon Gusto Ristorante	289
6	Daily Dose Midtown	287
7	Sakana Sushi & Grill	278
8	Good Fellas Grill	276
9	Stephano's Greek & Mediterranean Grill	275
10	Mandarin Oriental Tea Lounge	274

```
# view it by restaurant!
rest_star%>% group_by(name) %>% tally() %>% view()
lmstars_SI <- lm(starsBusiness ~starsReview, data =resReviewsData %>% filter(name=="Sugar & Ice"))
lmstars_Alle <- lm(starsBusiness ~starsReview, data =resReviewsData %>% filter(name=="Allegro"))
lmstars_momo <- lm(starsBusiness ~starsReview, data =resReviewsData %>% filter(name=="Momo Sushi"))
lmstars_innout <- lm(starsBusiness ~starsReview, data =resReviewsData %>% filter(name=="In-N-Out Burger"))
lmstars_thai <- lm(starsBusiness ~starsReview, data =resReviewsData %>% filter(name=="Thai House"))

summary(lmstars_Alle)
summary(lmstars_momo)
summary(lmstars_innout)
summary(lmstars_thai)
```

```
essentially perfect fit: summary may be unreliable
Call:
lm(formula = starsBusiness ~ starsReview, data = resReviewsData %>%
    filter(name == "Thai House"))

Residuals:
       Min         1Q     Median         3Q        Max
-1.698e-15 -2.060e-16 -2.060e-16  2.920e-16  5.922e-14

Coefficients:
              Estimate Std. Error    t value Pr(>|t|)
(Intercept)  4.000e+00  7.614e-16  5.254e+15  < 2e-16 ***
starsReview -4.974e-16  1.835e-16 -2.711e+00  0.00711 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.515e-15 on 292 degrees of freedom
Multiple R-squared:  0.4996,	Adjusted R-squared:  0.4979
F-statistic: 291.5 on 1 and 292 DF,  p-value: < 2.2e-16
```

Result: showing “Warning: essentially perfect fit: summary may be unreliable.

But the coefficients for starsReview are mostly not statistically significant since the sample size is not sufficient enough. So this attempt might not be an appropriate approach.

(B)

What are some words indicative of positive and negative sentiment? (One approach is to determine the average star rating for a word based on star ratings of documents where the word occurs). Do these 'positive' and 'negative' words make sense in the context of user reviews 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).

word <chr>	n <int>
food	32113
service	15844
time	12521
chicken	9411
restaurant	8833
nice	7907
menu	7574
love	7145
delicious	7090
bar	6202

- **Prune the words:**
  rareWords <-rrTokens %>% count(word, sort=TRUE) %>% filter(n<10)
  freqWords <-rrTokens %>% count(word, sort=TRUE) %>% filter(n>8000)

  Here is how I decided to prune the"feqWords" :
  - First, I observed the relatively even distribution among 5 stars for those words with n>8000, which can indicate these words do not possess distinct character for specific star review, they are words with neutral meaning.

    ws %>% filter(word==**'food'**) %>% view()
    ws %>% filter(word==**'service'**) %>% view()
    ws %>% filter(word==**'time'**) %>% view()
    ws %>% filter(word==**'chicken'**) %>% view()
    ws %>% filter(word==**'restaurant'**) %>% view()

	starsReview	word	n	prop
1	5	food	10827	0.02419858
2	4	food	7861	0.01884870
3	3	food	4952	0.02083833
4	1	food	4339	0.02404877
5	2	food	4134	0.02392915

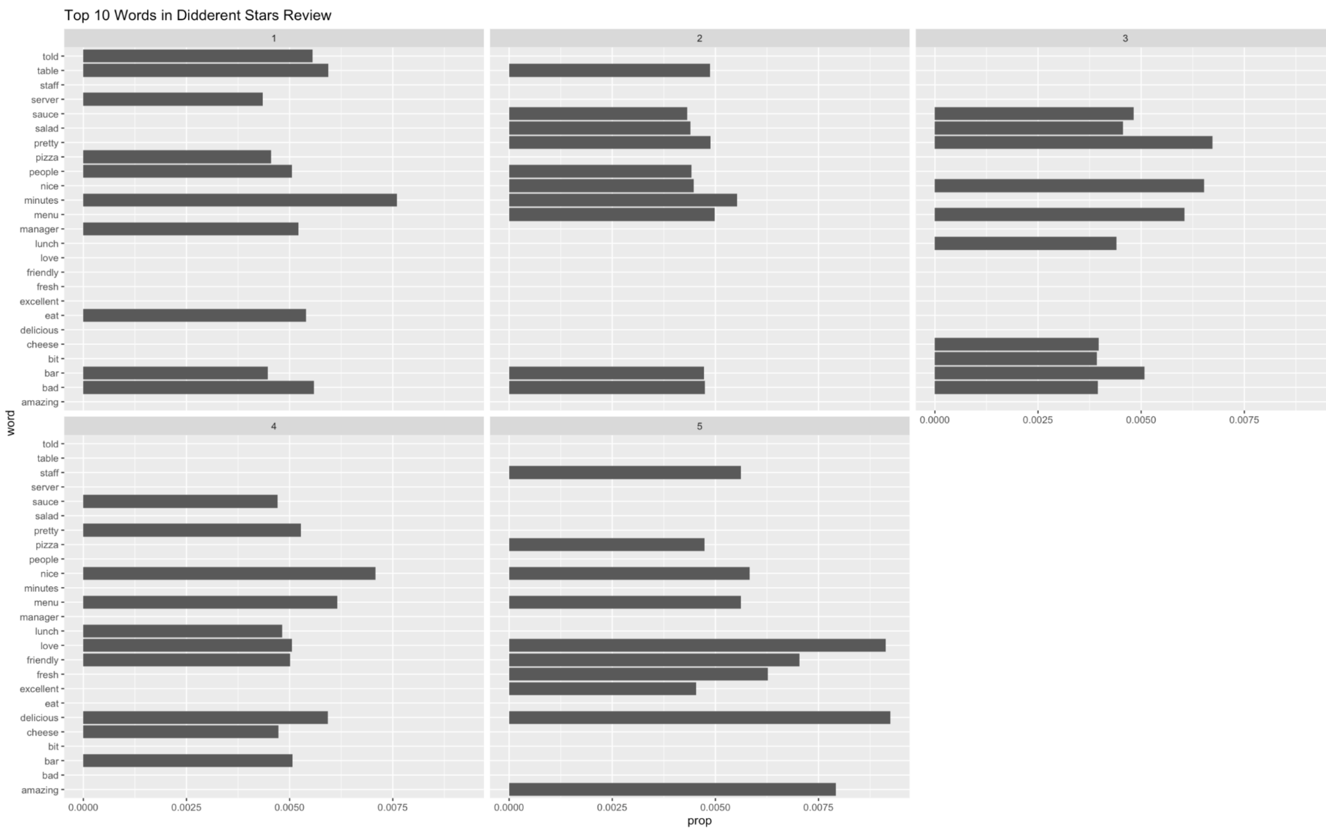
	starsReview	word	n	prop
1	5	service	5369	0.011999830
2	4	service	3955	0.009483093
3	3	service	2411	0.010145641
4	1	service	2131	0.011811002
5	2	service	1978	0.011449410

	starsReview	word	n	prop
1	5	time	3823	0.008544487
2	4	time	3333	0.007991694
3	3	time	1944	0.008180475
4	1	time	1843	0.010214771
5	2	time	1578	0.009134059

	starsReview	word	n	prop
1	5	chicken	3020	0.006749765
2	4	chicken	2729	0.006543454
3	3	chicken	1608	0.006766566
4	1	chicken	1030	0.005708743
5	2	chicken	1024	0.005927298

	starsReview	word	n	prop
1	5	restaurant	2922	0.006530733
2	4	restaurant	2210	0.005299023
3	3	restaurant	1304	0.005487315
4	1	restaurant	1279	0.007088818
5	2	restaurant	1118	0.006471405

Let's take a look at the Top 10 Words in each level of star reviews.



Regarding the separation of positive and negative reviews, I considered star 3 as a neutral review so I then classified star 4 and 5 as positive reviews, star 1 and 2 as negative.

After having a rough idea about the word distributions from the above images, I also looked at the top 10 frequent words for each star to point out which words indicate different star levels. Following my sentiment categories, I only extract the information from negative reviews, star 1 and 2, and positive reviews, star 4 and 5. I will explain more on the separation between negative and positive sentiment

	starsReview	word	n	prop
1	5	delicious	3897	0.0092463852
2	5	love	3848	0.0091301232
3	5	amazing	3339	0.0079224224
4	5	friendly	2965	0.0070350352
5	5	fresh	2645	0.0062757734
6	5	nice	2457	0.0058297071
7	5	staff	2367	0.0056161647
8	5	menu	2365	0.0056114193
9	5	pizza	1995	0.0047335228
10	5	excellent	1911	0.0045342166

	starsReview	word	n	prop
1	4	nice	2812	0.0070836587
2	4	menu	2441	0.0061490793
3	4	delicious	2355	0.0059324382
4	4	pretty	2092	0.0052699196
5	4	bar	2011	0.0050658740
6	4	love	2008	0.0050583167
7	4	friendly	1991	0.0050154924
8	4	lunch	1912	0.0048164849
9	4	cheese	1878	0.0047308361
10	4	sauce	1868	0.0047056453

	starsReview	word	n	prop
1	2	minutes	901	0.0055300501
2	2	menu	812	0.0049837965
3	2	pretty	796	0.0048855936
4	2	table	794	0.0048733183
5	2	bad	773	0.0047444270
6	2	bar	769	0.0047198763
7	2	nice	729	0.0044743690
8	2	people	720	0.0044191299
9	2	salad	717	0.0044007169
10	2	sauce	704	0.0043209270

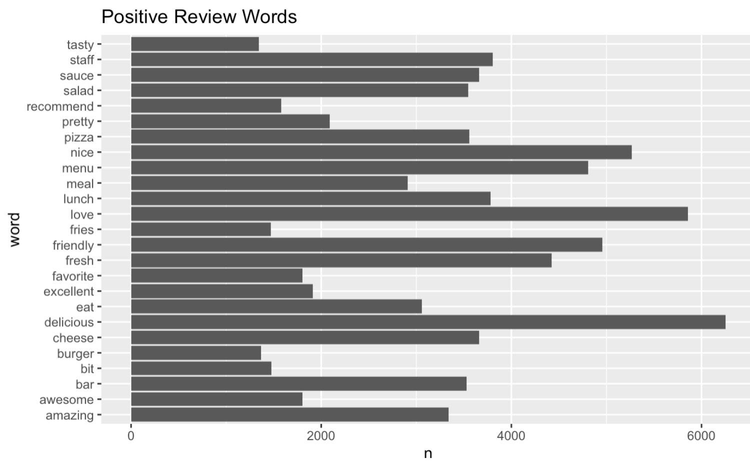
	starsReview	word	n	prop
1	1	minutes	1290	0.0075970389
2	1	table	1008	0.0059362909
3	1	bad	948	0.0055829402
4	1	told	943	0.0055534943
5	1	eat	916	0.0053944866
6	1	manager	885	0.0052119221
7	1	people	859	0.0050588034
8	1	pizza	772	0.0045464450
9	1	bar	759	0.0044698857
10	1	server	738	0.0043462130

Below are the top 20 words from positive (star 4 + star 5) and negative (star 1 + star 2) :

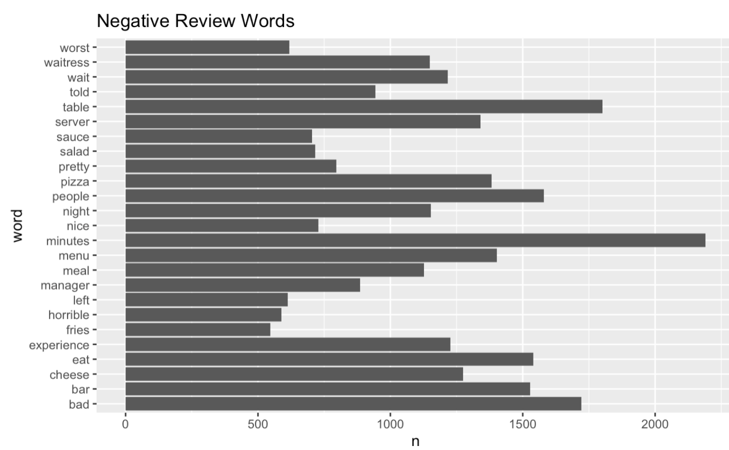
- ws %>% filter(starsReview==5| starsReview==4) %>% filter(row_number()<=20) %>% ggplot(aes(word, n)) + geom_col()+coord_flip()+ggtitle("**Positive** Review Words")

- ws %>% filter(starsReview==1| starsReview==2) %>% filter(row_number()<=20) %>% ggplot(aes(word, n)) + geom_col()+coord_flip()+ggtitle("**Negative** Review Words")

From my point of view, besides some noun phrases existing in both sentiments, such as staff and menu, words in positive reviews make sense.
We can infer that negative reviews focus more on accusing the services since there are more noun phrases, for instance, manager, people. This actually is understandable, because good service is something people expect to experience in a dining environment. In this case, customers will be less likely to praise the good quality of service but more likely to scold the bad one.
And in my knowledge, the term, pretty, serves as an adverb instead of an adjective, used as "pretty bad" etc.



starsReview <dbl>	word <chr>	n <int>	prop <dbl>
5	delicious	3897	0.009246385
5	love	3848	0.009130123
5	amazing	3339	0.007922422
4	nice	2812	0.007083659
5	friendly	2965	0.007035035
5	fresh	2645	0.006275773
4	menu	2441	0.006149079
4	delicious	2355	0.005932438
5	nice	2457	0.005829707
5	staff	2367	0.005616165



starsReview <dbl>	word <chr>	n <int>	prop <dbl>
1	minutes	1290	0.007597039
1	table	1008	0.005936291
1	bad	948	0.005582940
1	told	943	0.005553494
2	minutes	901	0.005530050
1	eat	916	0.005394487
1	manager	885	0.005211922
1	people	859	0.005058803
2	menu	812	0.004983797
2	pretty	796	0.004885594

- Using score, **starsReview*prop**, to find the top 20 words

```
# (c) score: starsReview*prop
```{r}
#Can we get a sense of which words are related to higher/lower star raings in general?
#One approach is to calculate the average star rating associated with each word - can sum the star ratings associated with reviews where each word occurs in.  Can consider the proportion of each word among reviews with a star rating.

xx<- ws %>% group_by(word) %>%summarise(totWS=sum(starsReview*prop))
# 20 words with highest and lowest star rating
xx %>% top_n(20)
xx %>% top_n(-20)

xx_neg<- ws %>% group_by(word) %>% filter(starsReview==1| starsReview==2) %>%  summarise(totWS=sum(starsReview*prop))
xx_pos<- ws %>% group_by(word) %>% filter(starsReview==4| starsReview==5) %>%  summarise(totWS=sum(starsReview*prop))

xx_neg %>% top_n(20)
xx_neg %>% top_n(-20)

xx_pos %>% top_n(20)
xx_pos %>% top_n(-20)
```
```

top_n(20) for positive and negative:

	word	totWS
1	delicious	0.06996168
2	love	0.06588388
3	nice	0.05748317
4	friendly	0.05523715
5	menu	0.05265341
6	amazing	0.04961791
7	fresh	0.04931473
8	staff	0.04254035
9	lunch	0.04143876
10	cheese	0.04009963
11	sauce	0.04007005
12	pizza	0.03943707
13	salad	0.03887131
14	bar	0.03826037
15	excellent	0.03400695
16	eat	0.03383253
17	meal	0.03203499
18	pretty	0.03179239
19	awesome	0.03083785
20	favorite	0.03041822

	word	totWS
1	minutes	0.01865714
2	table	0.01568293
3	bad	0.01507179
4	bar	0.01390964
5	people	0.01389706
6	menu	0.01344810
7	eat	0.01304204
8	salad	0.01218182
9	pizza	0.01204669
10	cheese	0.01196626
11	pretty	0.01190896
12	server	0.01173598
13	sauce	0.01155700
14	nice	0.01152231
15	told	0.01095465
16	experience	0.01085387
17	wait	0.01081364
18	night	0.01045639
19	waitress	0.01031788
20	meal	0.01007337

Comparing the results from above, using n and prop to extract top-ranked words is alike. Using both methods to validate these words are representable to a certain degree.

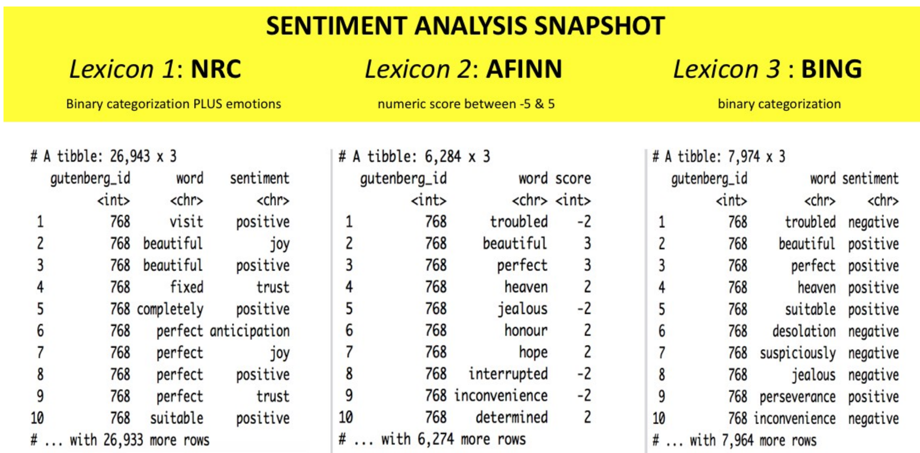
(C)

Digging deeper into Dictionaries -> How many matching terms are there for each of the dictionaries?

Using the dictionary based on positive and negative terms to predict sentiment of a movie: using 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.

Trying to predict review sentiment based on these aggregated scores, and evaluate their performance

Quick overview of the different dictionaries.



A tibble: 26,943 x 3

	gutenberg_id <int>	word <chr>	sentiment <chr>
1	768	visit	positive
2	768	beautiful	joy
3	768	beautiful	positive
4	768	fixed	trust
5	768	completely	positive
6	768	perfect	anticipation
7	768	perfect	joy
8	768	perfect	positive
9	768	perfect	trust
10	768	suitable	positive

... with 26,933 more rows

A tibble: 6,284 x 3

	gutenberg_id <int>	word <chr>	score <int>
1	768	troubled	-2
2	768	beautiful	3
3	768	perfect	3
4	768	heaven	2
5	768	jealous	-2
6	768	honour	2
7	768	hope	2
8	768	interrupted	-2
9	768	inconvenience	-2
10	768	determined	2

... with 6,274 more rows

A tibble: 7,974 x 3

	gutenberg_id <int>	word <chr>	sentiment <chr>
1	768	troubled	negative
2	768	beautiful	positive
3	768	perfect	positive
4	768	heaven	positive
5	768	suitable	positive
6	768	desolation	negative
7	768	suspiciously	negative
8	768	jealous	negative
9	768	perseverance	positive
10	768	inconvenience	negative

... with 7,964 more rows

Matching in Bing and Affin

- **Using Bing**

```
#to retain only the words which match the sentiment dictionary, do an inner-join
rrSenti_bing<- rrTokens %>% inner_join( get_sentiments("bing"), by="word")

#  Which words contribute to positive/negative sentiment ?
#count the occurrences of positive/negative sentiment words in the reviews
xx<-rrSenti_bing %>% group_by(word, sentiment) %>% summarise(totOcc=sum(n)) %>% arrange(sentiment,
desc(totOcc))
xx

xx_neg<-rrSenti_bing %>% group_by(word, sentiment) %>%filter(sentiment=="negative") %>%
summarise(totOcc=sum(n)) %>% arrange(sentiment, desc(totOcc))
xx_pos<-rrSenti_bing %>% group_by(word, sentiment) %>%filter(sentiment=="positive") %>%
summarise(totOcc=sum(n)) %>% arrange(sentiment, desc(totOcc))

xx_neg
xx_pos
```

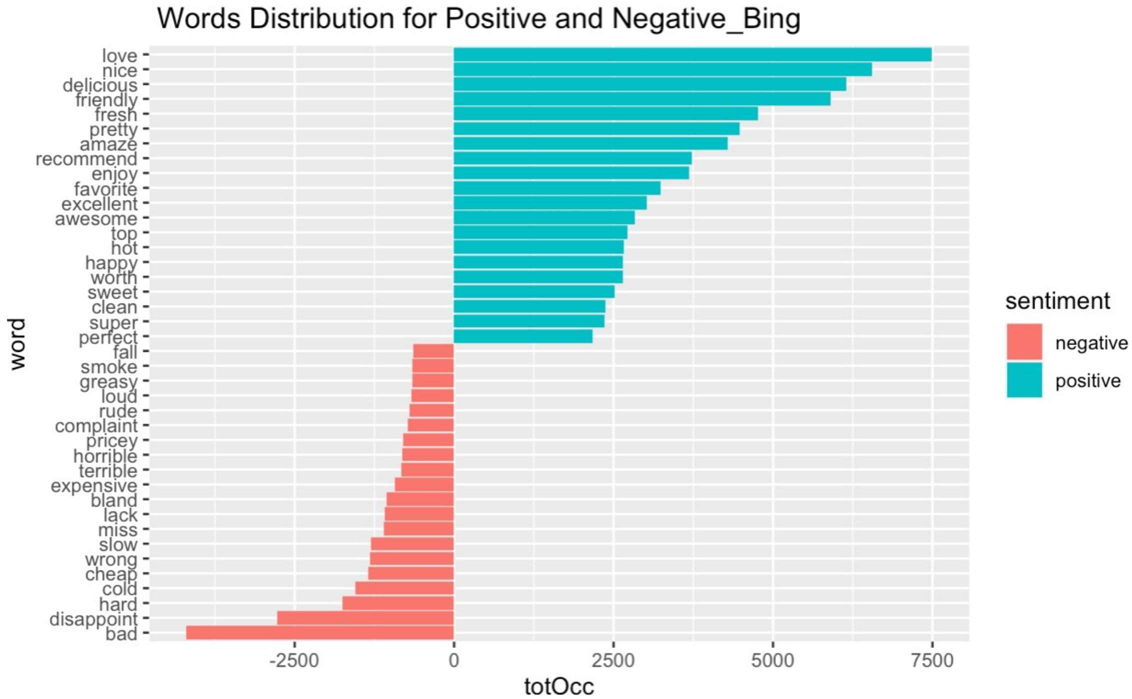
- Negative words defined by Bing that occur in the review data

word <chr>	sentiment <chr>	totOcc <int>
bad	negative	4199
disappoint	negative	2773
hard	negative	1750
cold	negative	1542
cheap	negative	1349
wrong	negative	1324
slow	negative	1296
miss	negative	1099
lack	negative	1093
bland	negative	1062

- Positive words defined by Bing that occur in the review data

word <chr>	sentiment <chr>	totOcc <int>
love	positive	7492
nice	positive	6554
delicious	positive	6143
friendly	positive	5902
fresh	positive	4767
pretty	positive	4480
amaze	positive	4294
recommend	positive	3723
enjoy	positive	3686
favorite	positive	3240

⬇️ From below plot we can tell that using totOcc (total occurance) to capture the terms in different levels is making sense. Not too many confusing terms or meaningless noun phrases selected.



- **1130** matching words with Bing dictionary

```
### (c) How many matching terms are there for each of the dictionaries?

*For the bing dictionary we found a total of 1130 total matching words*
```{r warning=FALSE}
bingMatchUniqueWords <- unique(rrSenti_bing$word)
str(bingMatchUniqueWords)
```
```

```
 chr [1:1130] "fresh" "friendly" "fun" "ready" "enjoyable" "hard" "memorable" "negative" "fast" "happy" "improve" "pan" "amaze" "easy" ..
```

- Besides part c, I would like to sort the words using " tf * idf ".
  Note: tf * idf can somehow represent the importance of words.
  - avg=mean(tf_idf)
  -

  From the below result, although it might seem fair, the high ranking words actually have no-so-high total occurrence. This might not be persuasive to represent those words for Positive or Negative reviews.

```
xx<-rrSenti_bing %>% group_by(word, sentiment) %>% summarise(avg=mean(tf_idf),totOcc=sum(n)) %>% arrange(sentiment, desc(avg))
xx

xx_neg_imp<-rrSenti_bing %>% group_by(word, sentiment) %>% filter(sentiment=="negative") %>% summarise(avg=mean(tf_idf),totOcc=sum(n)) %>%
arrange(sentiment, desc(avg))
xx_pos_imp<-rrSenti_bing %>% group_by(word, sentiment) %>% filter(sentiment=="positive") %>% summarise(avg=mean(tf_idf),totOcc=sum(n)) %>%
arrange(sentiment, desc(avg))
```

word <chr>	sentiment <chr>	avg <dbl>	totOcc <int>
awsome	positive	0.50605052	27
handsome	positive	0.47041971	15
pep	positive	0.44768222	11
approve	positive	0.38743708	34
speedy	positive	0.38116752	56
awesomeness	positive	0.37838367	18
reliable	positive	0.37441650	41
carefree	positive	0.36700868	13
splendid	positive	0.34934713	14
serene	positive	0.34250639	15

word <chr>	sentiment <chr>	avg <dbl>	totOcc <int>
forgetful	negative	0.47792862	10
disregard	negative	0.45639741	10
gimmicky	negative	0.42809842	11
inefficient	negative	0.42177783	9
pinch	negative	0.40929484	28
violation	negative	0.40704119	9
dissapointed	negative	0.39207663	34
cranky	negative	0.38308987	11
abysmal	negative	0.37949125	14
spendy	negative	0.37516338	14

- **Accuracy for Bing: 83.8%**

```
#consider reviews with 1 to 2 stars as positive, and this with 4 to 5 stars as negative
revSenti_bing <- revSenti_bing %>% mutate(hiLo=ifelse(starsReview<=2,-1, ifelse(starsReview>=4, 1, 0 )))
revSenti_bing <- revSenti_bing %>% mutate(pred_hiLo=ifelse(sentiScore >0, 1, -1))

#filter out the reviews with 3 stars, and get the confusion matrix for hiLo vs pred_hiLo
xx<-revSenti_bing %>% filter(hiLo!=0)
table(actual=xx$hiLo, predicted=xx$pred_hiLo )

#accuracy
BingAccuracy <-mean(xx$hiLo==xx$pred_hiLo)
```

```
       predicted
actual    -1     1
    -1  6362  2001
    1   3441 21793
```

To see how the words in Bing are presentable or not:
My answer will be Yes, it is representative, because we can tell from the ascending average sentimental score (avgSentiSc) from star 1 to 5.

starsReview <dbl>	avgPos <dbl>	avgNeg <dbl>	avgSentiSc <dbl>
1	0.3107697	0.6892303	-0.3784607
2	0.4483663	0.5516337	-0.1032674
3	0.6104126	0.3895874	0.2208253
4	0.7550768	0.2449232	0.5101536
5	0.8322997	0.1677003	0.6645993

- **Afinn: 84.16%**

```
#we can consider reviews with 1 to 2 stars as positive, and this with 4 to 5 stars as negative
revSenti_afinn <- revSenti_afinn %>% mutate(hiLo=ifelse(starsReview<=2,-1, ifelse(starsReview>=4, 1, 0 )))
revSenti_afinn <- revSenti_afinn %>% mutate(pred_hiLo=ifelse(sentiSum >0, 1, -1))
#filter out the reviews with 3 stars, and get the confusion matrix for hiLo vs pred_hiLo
xx<-revSenti_afinn %>% filter(hiLo!=0)
table(actual=xx$hiLo, predicted=xx$pred_hiLo )
#accuracy
AffinAccuracy <-mean(xx$hiLo==xx$pred_hiLo)
```

```
          predicted
actual     -1     1
      -1  5117  3081
      1   2132 22572
```

To see how the words in Afinn are presentable or not:

-Method 1:

>> revSenti_afinn <- rrSenti_afinn %>% group_by(review_id, starsReview) %>% summarise(nwords=n(), sentiSum =sum(value))

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

🙌The result makes sense

starsReview <dbl>	avgLen <dbl>	avgSenti <dbl>
1	4.994694	-2.3863899
2	5.051773	0.7765985
3	4.983273	3.7701958
4	4.870839	6.5102021
5	4.343821	7.2800636

Method 2:

I experimentally defined the goodBad for “affin score -5 to -3” as minus, and for “affin score 3 to 5” as positive score, and among -2 to 2 will be zero.

>> rrSenti_afinn <- rrSenti_afinn %>% mutate(goodBad=ifelse(value %in% c('-5', '-4', '-3'), -totOcc, ifelse(value %in% c('5', '4', '3'), totOcc, 0)))

>> rrSenti_afinn %>% group_by(starsReview) %>% summarise(avgscore=sum(goodBad))

🙌 And the result is making sense too.

starsReview	avgscore
<dbl>	<dbl>
1	-10890
2	8319
3	30241
4	91929
5	126477

Afinn: 84.16%

```
      predicted
actual    -1     1
    -1  5117  3081
    1   2132 22572
```


starsReview	avgscore
<dbl>	<dbl>
1	-10890
2	8319
3	30241
4	91929
5	126477

Summary for part C:

The accuracy of prediction for Bing and Affin are higher than nrc and are almost equal to each other. but from the results, the extracted words from Affin make more sense compared to Bing But for part D, I will pick Bing to conduct further analysis since it has embedded categories, positive and negative, in the dictionary, and I would love to take a deeper look about how this default categorize system works in predicting the hiLo score.

(D) Model-developing part to predict reviews sentiment.

(i)

Here I will go for Lemmatizing. Since I would like to capture more meaning than having a reduced number set of terms.

Do you use term frequency, tfidf, or other measures, and why? What is the size of the document- term matrix?

Should you use stemming or lemmatization when using the dictionaries?

Ans:

I will use lemmatization because I want to put more emphasis on the context inthe reviews instead of having a compact number of terms. And to my understanding lemmatization conducts a better work on capturing the meaning of terms while stemming is good at reducing the size of unique words.

This chart presents a straight overview of the results from RF and SVM models. In the following model building process, I use hiLo score to predict the outcome.

The definition of “**hiLo**” is
➡️ hiLo=ifelse(starsReview<=2,-1, ifelse(starsReview>=4, 1, 0)

The definition of “predicted hiLo” is
➡️ pred_hiLo=ifelse(sentiScore >0, 1, -1)
sentiScore = posProp - negProp,
posProp=posSum/nwords
negProp=negSum/nwords

Notes: In order to do some comparison, I put efforts on building 2 models, RF and SVM.

	RF_Bing	**SVM_Bing**
Training	0.9710476	0.9757143
Testing	0.8822222	0.8833333

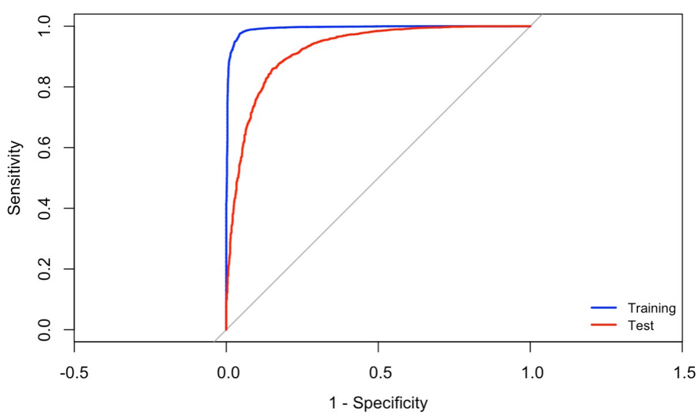
(i) Using Bing to build a RF

Use a subset of 15,000 samples.
0.7 as training
0.3 as testing
Using ROC can find the best Threshold: 0.5995781
Set seed to make sure we get the same testing and training data set everytime we run it.

Top16 Var important:

```
         bad    delicious     terrible     horrible         love        bland     friendly         rude
1.255137e-02 1.046832e-02 8.981699e-03 8.694064e-03 6.822255e-03 6.264299e-03 6.126269e-03 5.803121e-03
        poor        amaze        awful    excellent     favorite      awesome     mediocre   disappoint
5.361846e-03 5.211219e-03 5.094798e-03 4.748572e-03 4.051279e-03 4.047077e-03 3.623737e-03 3.611445e-03
```

Plot the ROC for Training and Testing data.



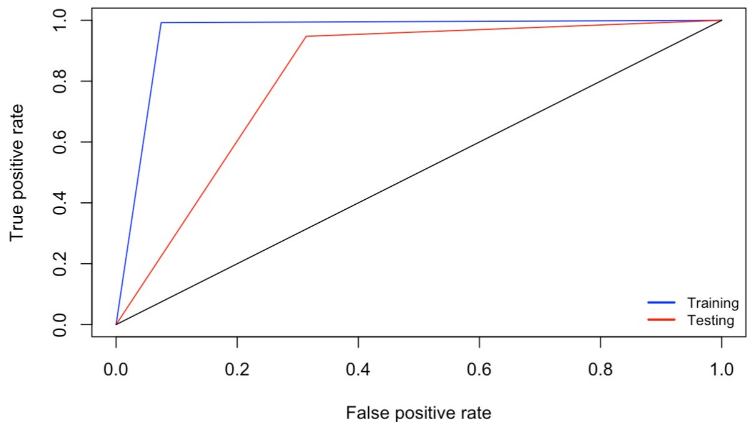
Using Bing to build a SVM

```
svmBing2 <- svm(as.factor(hiLo) ~., data = revDTM_sentiBing_trn
%>% select(-review_id), kernel="radial", cost=5, gamma=5, scale=FALSE)
```

```
revDTM_predTrn_svm2Bing<-predict(svmBing2, revDTM_sentiBing_trn)
table(actual= revDTM_sentiBing_trn$hiLo, predicted= revDTM_predTrn_svm2Bing)
revDTM_predTst_svm2Bing<-predict(svmBing2, revDTM_sentiBing_tst)
table(actual= revDTM_sentiBing_tst$hiLo, predicted= revDTM_predTst_svm2Bing)

svm_bing_acc_tr <- mean(revDTM_sentiBing_trn$hiLo == revDTM_predTrn_svm2Bing) # 0.9757143
svm_bing_acc_ts <- mean(revDTM_sentiBing_tst$hiLo == revDTM_predTst_svm2Bing) # 0.8833333
```

Plot the ROC for Training and Testing data.



(ii)

Develop a model using a broader list of terms. And I will compare performance with that in part (c) above

For the previous dataset, we exempt rare words(n<10) and frequent words(n>8000).
And here we revised those standards to n>100 and n<6000 , with sample size=15,000, to prevent expensive computation issues.

As to stemming or lemmatizing, I will, still, use lemmatizing over stemming since I still want to put more emphasis on analyzing the "context" in the reviews.

In (c), the accuracy of Bing dictionary to predict sentiment using scores is ➡️ 83.8%
> posSum=sum(sentiment=='positive')
> negSum=sum(sentiment=='negative')

	RF_Bing	**RF_broader**
Training	0.9710476	1.0
Testing	0.8822222	1.0
	SVM_Bing	**SVM_broader**
Training	0.9757143	1.0
Testing	0.8833333	Run into unsolvable error

Notes: I have run into some bizarre error in my code (line 878-890), saying "testing set and model are not match". But I have double checked all the splitting steps and model-building set. Error still occurs.

(E)

Since this dataset has different attributes for restaurants. I will 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.

Using the model I developed previously, I want to see whether prediction accuracy vary by certain restaurant attributes.

For the attributes of my choice, I chose “GoodForKids” and “RestaurantsReservations”.
The main question I was asking myself in the picking process is what factors will affect the customers to leave more distinguished or extreme reviews. If the content of reviews is more evident in positive or negative sentiment, it is easier for models to capture the essence and further conduct better predictions.

Concerning “GoodForKids”, my thought on these attributes is that restaurants that are labeled "True" or "False" must be put on by customers with children. So this factor is highly relevant to the reviewers. So we might be able to extract some insights through analyzing this attribute.

In the matter of “RestaurantsReservations”, I assume that if people make reservations for the restaurant, they might possess higher expectations than not making reservations. If the reality appeared not to match their hype, they would give even lower stars for the restaurant. And from this pattern, we might be able to get some insights too.

- “Good For Kids” or not

GoodForKids <chr>	n <int>
False	8804
True	30621
NA	539

GoodForKids <chr>	avgStar <dbl>
False	3.597342
True	3.718886
NA	3.914657

- “Restaurants Reservations”

RestaurantsReservations <chr>	n <int>
False	20949
True	18352
NA	663

RestaurantsReservations <chr>	avgStar <dbl>
False	3.645663
True	3.744823
NA	3.859729

From the above results, we can conclude that they do showcase a difference between “True” and ”False”.

While for the “GoodForKids” attributes, I do not expect higher star reviews on the NA label, since my assumption is that people who leave a “True” label on the factor might leave a higher rating for the restaurant since the staff there are good with their kids. If I am the stakeholder of this research, I will conduct further steps to gather more information about this observed phenomenon.

Steps I took to conduct this analysis:

- **Step 1:**
  Separate data set into ”attributes of your choice” == ”True” and ==”False”. And extract sets of “review_id”
- **Step 2:**
  Create 2 separate dataset to build models
- **Step 3:**
  Compare the accuracy between models with “True” or “False” labels.

Since I picked 2 attributes, GoodForKids and RestaurantsReservations, I will repeat the above steps twice.

I will use “Random Forest” with terms from the Bing dictionary. And you might ask why don’t I choose the SVM model, it is because although SVM has a slightly higher accuracy than Bing on the training and testing data, RF has a smaller difference between training and testing.

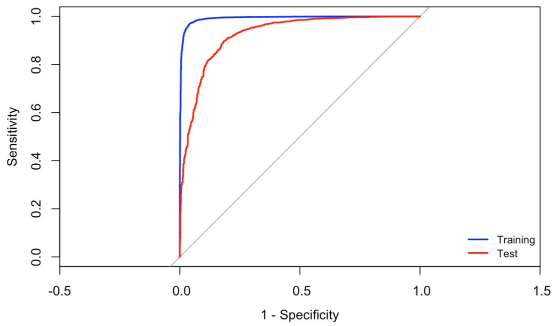
Using ROC find the **Threshold** for 4 different attributes conditions

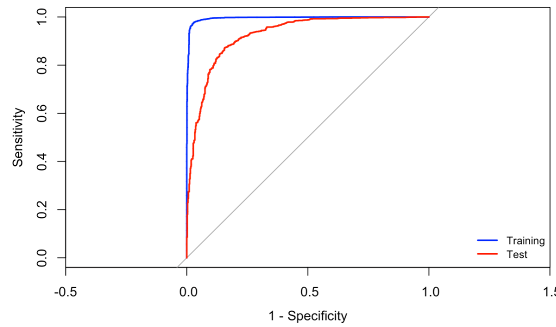
- GoodForKids(True): **0.6244062**
- GoodForKids(False)**: 0.6282966**
- Restaurants Reservations(True): **0.6459355**
- Restaurants Reservations(False): **0.5888654**

And here are the accuracy for different attributes conditions:

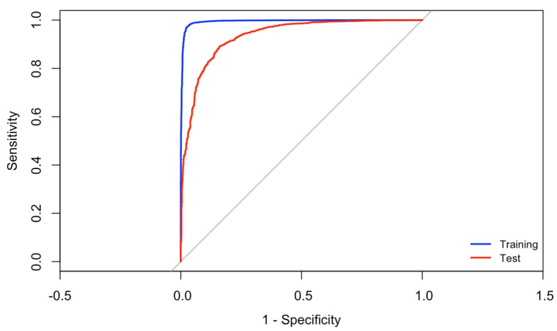
	GoodForKids==”True”	**GoodForKids==”False”**
Training	0.9671429	0.9755102
Testing	0.8851111	0.8811111
	RestaurantsReservations==”True”	**RestaurantsReservations==”False”**
Training	0.9729524	0.9712245
Testing	0.8882222	0.8738095

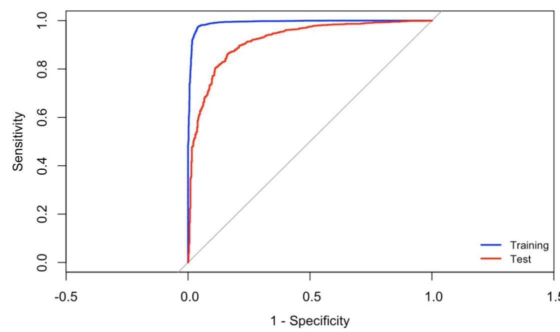
- GoodForKids: True (left) vs False (right)





- RestaurantsReservations : True (left) vs False (right)





From the above chart and images, we can conclude that these two attributes do not significantly affect the accuracy rates. The results are basically compared to part D. And I would love to get some feedback about my approach for part E, and further on experiment more on different attributes.