

# ASSIGNMENT 3 (IDS 572 DATA MINING FOR BUSINESS)

*Aarjav Sanghvi*

*Shruti Chanda*

*Shubham Chaudhary*

## ABSTRACT

This assignment performs text mining techniques and sentiment analysis on Yelp reviews. We would analyse the user's review of restaurants and their sentiments behind the star ratings and thereby predicting a pattern between sentiments and ratings, review and sentiments and how various text dictionaries like – Bing, NRC and AFINN help predict the sentiment of a review. While performing these tasks we'll try and assess data for answering various questions.

# QUESTIONS

### Q1. Explore the data.

Ans. Review of the original data:

Distinct words:

```
> #distinct words  
> dim(rrTokens %>% distinct(word))  
[1] 43941      1  
> |
```

```
47 #distinct words
48 rFTokens %%> distinct(word)
49
50 PART 1:
51
Console Terminal Jobs x
R 4.1.1 - D:/MSBA/1stSem/DMB/Assignment/3rd
2 came
3 here
4 or
5 dinner
6 to
7 celebrate
8 my
9 friends
10 birthday
11 the
12 restaurant
13 itself
14 is
15 beautiful
16 and
17 service
18 incredible
19 but
20 reason
21 reason
22 felt
23 like
24 something
25 was
26 missing
27 food
28 decent
29 way
30 ever
31 priced
32 guess
33 money
34 mind
35 and
36 expecting
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6040
6051
6052
6053
6054
6055
6056
6057
6058
6059
6050
6061
6062
6063
6064
6065
6066
6067
6068
6069
6060
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6070
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6080
6090
6091
6092
6093
6094
6095
6096
6097
6098
6099
6090
60100
60101
60102
60103
60104
60105
60106
60107
60108
60109
60110
60111
60112
60113
60114
60115
60116
60117
60118
60119
60110
60120
60121
60122
60123
60124
60125
60126
60127
60128
60129
60110
60130
60131
60132
60133
60134
60135
60136
60137
60138
60139
60110
60140
60141
60142
60143
60144
60145
60146
60147
60148
60149
60110
60150
60151
60152
60153
60154
60155
60156
60157
60158
60159
60110
60160
60161
60162
60163
60164
60165
60166
60167
60168
60169
60110
60170
60171
60172
60173
60174
60175
60176
60177
60178
60179
60110
60180
60181
60182
60183
60184
60185
60186
60187
60188
60189
60110
60190
60191
60192
60193
60194
60195
60196
60197
60198
60199
60110
60200
60201
60202
60203
60204
60205
60206
60207
60208
60209
60210
60211
60212
60213
60214
60215
60216
60217
60218
60219
60210
60220
60221
60222
60223
60224
60225
60226
60227
60228
60229
60210
60230
60231
60232
60233
60234
60235
60236
60237
60238
60239
60210
60240
60241
60242
60243
60244
60245
60246
60247
60248
60249
60210
60250
60251
60252
60253
60254
60255
60256
60257
60258
60259
60210
60260
60261
60262
60263
60264
60265
60266
60267
60268
60269
60210
60270
60271
60272
60273
60274
60275
60276
60277
60278
60279
60210
60280
60281
60282
60283
60284
60285
60286
60287
60288
60289
60210
60290
60291
60292
60293
60294
60295
60296
60297
60298
60299
60210
60300
60301
60302
60303
60304
60305
60306
60307
60308
60309
60310
60311
60312
60313
60314
60315
60316
60317
60318
60319
60310
60320
60321
60322
60323
60324
60325
60326
60327
60328
60329
60310
60330
60331
60332
60333
60334
60335
60336
60337
60338
60339
60310
60340
60341
60342
60343
60344
60345
60346
60347
60348
60349
60310
60350
60351
60352
60353
60354
60355
60356
60357
60358
60359
60310
60360
60361
60362
60363
60364
60365
60366
60367
60368
60369
60310
60370
60371
60372
60373
60374
60375
60376
60377
60378
60379
60310
60380
60381
60382
60383
60384
60385
60386
60387
60388
60389
60310
60390
60391
60392
60393
60394
60395
60396
60397
60398
60399
60310
60400
60401
60402
60403
60404
60405
60406
60407
60408
60409
60410
60411
60412
60413
60414
60415
60416
60417
60418
60419
60410
60420
60421
60422
60423
60424
60425
60426
60427
60428
60429
60410
60430
60431
60432
60433
60434
60435
60436
60437
60438
60439
60410
60440
60441
60442
60443
60444
60445
60446
60447
60448
60449
60410
60450
60451
60452
60453
60454
60455
60456
60457
60458
60459
60410
60460
60461
60462
60463
60464
60465
60466
60467
60468
60469
60410
60470
60471
60472
60473
60474
60475
60476
60477
60478
60479
60410
60480
60481
60482
60483
60484
60485
60486
60487
60488
60489
60410
60490
60491
60492
60493
60494
60495
60496
60497
60498
60499
60410
60500
60501
60502
60503
60504
60505
60506
60507
60508
60509
60510
60511
60512
60513
60514
60515
60516
60517
60518
60519
60510
60520
60521
60522
60523
60524
60525
60526
60527
60528
60529
60510
60530
60531
60532
60533
60534
60535
60536
60537
60538
60539
60510
60540
60541
60542
60543
60544
60545
60546
60547
60548
60549
60510
60550
60551
60552
60553
60554
60555
60556
60557
60558
60559
60510
60560
60561
60562
60563
60564
60565
60566
60567
60568
60569
60510
60570
60571
60572
60573
60574
60575
60576
60577
60578
60579
60510
60580
60581
60582
60583
60584
60585
60586
60587
60588
60589
60510
60590
60591
60592
60593
60594
60595
60596
60597
60598
60599
60510
60600
60601
60602
60603
60604
60605
60606
60607
60608
60609
60610
60611
60612
60613
60614
60615
60616
60617
60618
60619
60610
60620
60621
60622
60623
60624
60625
60626
60627
60628
60629
60610
60630
60631
60632
60633
60634
60635
60636
60637
60638
60639
60610
60640
60641
60642
60643
60644
60645
60646
60647
60648
60649
60610
60650
60651
60652
60653
60654
60655
60656
60657
60658
60659
60610
60660
60661
60662
60663
60664
60665
60666
60667
60668
60669
60610
60670
60671
60672
60673
60674
60675
60676
60677
60678
60679
60610
60680
60681
60682
60683
60684
60685
60686
60687
60688
60689
60610
60690
60691
60692
60693
60694
60695
60696
60697
60698
60699
60610
60700
60701
60702
60703
60704
60705
60706
60707
60708
60709
60710
60711
60712
60713
60714
60715
60716
60717
60718
60719
60710
60720
60721
60722
60723
60724
60725
60726
60727
60728
60729
60710
60730
60731
60732
60733
60734
60735
60736
60737
60738
60739
60710
60740
60741
60742
60743
60744
60745
60746
60747
60748
60749
60710
60750
60751
60752
60753
60754
60755
60756
60757
60758
60759
60710
60760
60761
60762
60763
60764
60765
60766
60767
60768
60769
60710
60770
60771
60772
60773
60774
60775
60776
60777
60778
60779
60710
60780
60781
60782
60783
60784
60785
60786
60787
60788
60789
60710
60790
60791
60792
60793
60794
60795
60796
60797
60798
60799
60710
60800
60801
60802
60803
60804
60805
60806
60807
60808
60809
60810
60811
60812
60813
60814
60815
60816
60817
60818
60819
60810
60820
60821
60822
60823
60824
60825
60826
60827
60828
60829
60810
60830
60831
60832
60833
60834
60835
60836
60837
60838
60839
60810
60840
60841
60842
60843
60844
60845
60846
60847
60848
60849
60810
60850
60851
60852
60853
60854
60855
60856
60857
60858
60859
60810
60860
60861
60862
60863
60864
60865
60866
60867
60868
60869
60810
60870
60871
60872
60873
60874
60875
60876
60877
60878
60879
60810
60880
60881
60882
60883
60884
60885
60886
60887
60888
60889
60810
60890
60891
60892
60893
60894
60895
60896
60897
60898
60899
60810
60900
60901
60902
60903
60904
60905
60906
60907
60908
60909
60910
60911
60912
60913
60914
60915
60916
60917
60918
60919
60910
60920
60921
60922
60923
60924
60925
60926
60927
60928
60929
60910
60930
60931
60932
60933
60934
60935
60936
60937
60938
60939
60910
60940
60941
60942
60943
60944
60945
60946
60947
60948
60949
60910
60950
60951
60952
60953
60954
60955
60956
60957
60958
60959
60910
60960
60961
60962
60963
60964
60965
60966
60967
60968
60969
60910
60970
60971
60972
60973
60974
60975
60976
60977
60978
60979
60910
60980
60981
60982
60983
60984
60985
60986
60987
60988
60989
60910
60990
60991
60992
60993
60994
60995
60996
60997
60998
60999
60910
61000
61001
61002
61003
61004
61005
61006
61007
61008
61009
61010
61011
61012
61013
61014
61015
61016
61017
61018
61019
61010
61020
61021
61022
61023
61024
61025
61026
61027
61028
61029
61010
61030
61031
61032
61033
61034
61035
61036
61037
61038
61039
61010
61040
61041
61042
61043
61044
61045
61046
61047
61048
61049
61010
61050
61051
61052
61053
61054
61055
61056
61057
61058
61059
61010
61060
61061
61062
61063
61064
61065
61066
61067
61068
61069
61010
61070
61071
61072
61073
61074
61075
61076
61077
61078
61079
61010
61080
61081
61082
61083
61084
61085
61086
61087
61088
61089
61010
61090
61091
61092
61093
61094
61095
61096
61097
61098
61099
61010
61100
61101
61102
61103
61104
61105
61106
61107
61108
61109
61110
61111
61112
61113
61114
61115
61116
61117
61118
61119
61110
61120
61121
61122
61123
61124
61125
61126
61127
61128
61129
61110
61130
61131
61132
61133
61134
61135
61136
61137
61138
61139
61110
61140
61141
61142
61143
61144
61145
61146
61147
61148
61149
61110
61150
61151
61152
61153
61154
61155
61156
61157
61158
61159
61110
61160
61161
61162
61163
61164
61165
61166
61167
61168
61169
61110
61170
61171
61172
61173
61174
61175
61176
61177
61178
61179
61110
61180
61181
61182
61183
61184
61185
61186
61187
61188
61189
61110
61190
61191
61192
61193
61194
61195
61196
61197
61198
61199
61110
61200
61201
61202
61203
61204
61205
61206
61207
61208
61209
61210
61211
61212
61213
61214
61215
61216
61217
61218
61219
61210
61220
61221
61222
61223
61224
61225
61226
61227
61228
61229
61210
61230
61231
61232
61233
61234
61235
61236
61237
61238
61239
61210
61240
61241
61242
61243
61244
61245
61246
61247
61248
61249
61210
61250
61251
61252
61253
61254
61255
61256
61257
61258
61259
61210
61260
61261
61262
61263
61264
61265
61266
61267
61268
61269
61210
61270
61271
61272
61273
61274
61275
61276
61277
61278
61279
61210
61280
61281
61282
61283
61284
61285
61286
61287
61288
61289
61210
61290
61291
61292
61293
61294
61295
61296
61297
61298
61299
61210
61300
61301
61302
61303
61304
61305
61306
61307
61308
61309
61310
61311
61312
61313
61314
61315
61316
61317
61318
61319
61310
61320
61321
61322
61323
61324
61325
61326
61327
61328
61329
61310
61330
61331
61332
61333
61334
61335
61336
61337
61338
61339
61310
61340
61341
61342
61343
61344
61345
61346
61347
61348
61349
61310
61350
61351
61352
61353
61354
61355
61356
61357
61358
61359
61310
61360
61361
61362
61363
61364
61365
61366
61367
61368
61369
61310
61370
61371
61372
61373
61374
61375
61376
61377
61378
61379
61310
61380
61381
61382
61383
61384
61385
61386
61387
61388
61389
61310
61390
61391
61392
61393
61394
61395
61396
61397
61398
61399
61310
61400
61401
61402
61403
61404
61405
61406
61407
61408
61409
61410
61411
61412
61413
61414
61415
61416
61417
61418
61419
61410
61420
61421
61422
61423
61424
61425
61426
61427
61428
61429
61410
61430
61431
61432
61433
61434
61435
61436
61437
61438
61439
61410
61440
61441
61442
61443
61444
61445
61446
61447
61448
61449
61410
61450
61451
61452
61453
61454
61455
61456
61457
61458
61459
61410
61460
61461
61462
61463
61464
61465
61466
61467
61468
61469
61410
61470
61471
61472
61473
61474
61475
61476
61477
61478
61479
61410
61480
61481
61482
61483
61484
61485
61486
61487
61488
61489
61410
61490
61491
61492
61493
61494
61495
61496
61497
61498
61499
61410
61500
61501
61502
61503
61504
61505
61506
61507
61508
61509
61510
61511
61512
61513
61514
61515
61516
61517
61518
61519
61510
61520
61521
61522
61523
61524
61525
61526
61527
61528
61529
61510
61530
61531
61532
61533
61534
61535
61536
61537
61538
61539
61510
61540
61541
61542
61543
61544
61545
61546
61547
6
```

Stop words (using tidytext library):

```
> # remove stopwords  
> rrTokens <- rrTokens %>% anti_join(stop_words)  
Joining, by = "word"  
> dim(rrTokens)  
[1] 1572586      12  
>
```

## Frequency of words and sorting

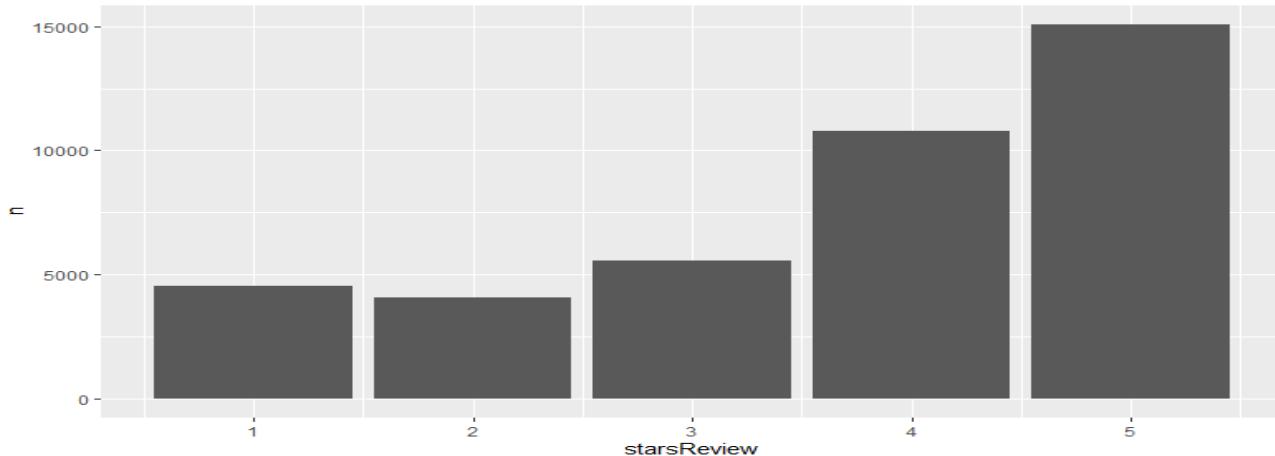
Removing rare words with frequency less than 50:

Removing words with digits:

```
> # remove the terms containing digits
> rrdf <- rrdf %>% filter(str_detect(word, "[0-9]") == FALSE)
> dim(rrdf)
[1] 1332609     12
> # remaining distinct tokens
> rrdf %>% distinct(word) %>% dim()
[1] 3523      1
> |
```

- i) How are star ratings distributed? How will you use the star ratings to obtain a label indicating ‘positive’ or ‘negative’ – explain using the data, graphs, etc.? Do star ratings have any relation to ‘funny’, ‘cool’, ‘useful’? Is this what you expected?
  - ii) 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?)

Ans. i) Star ratings distribution:

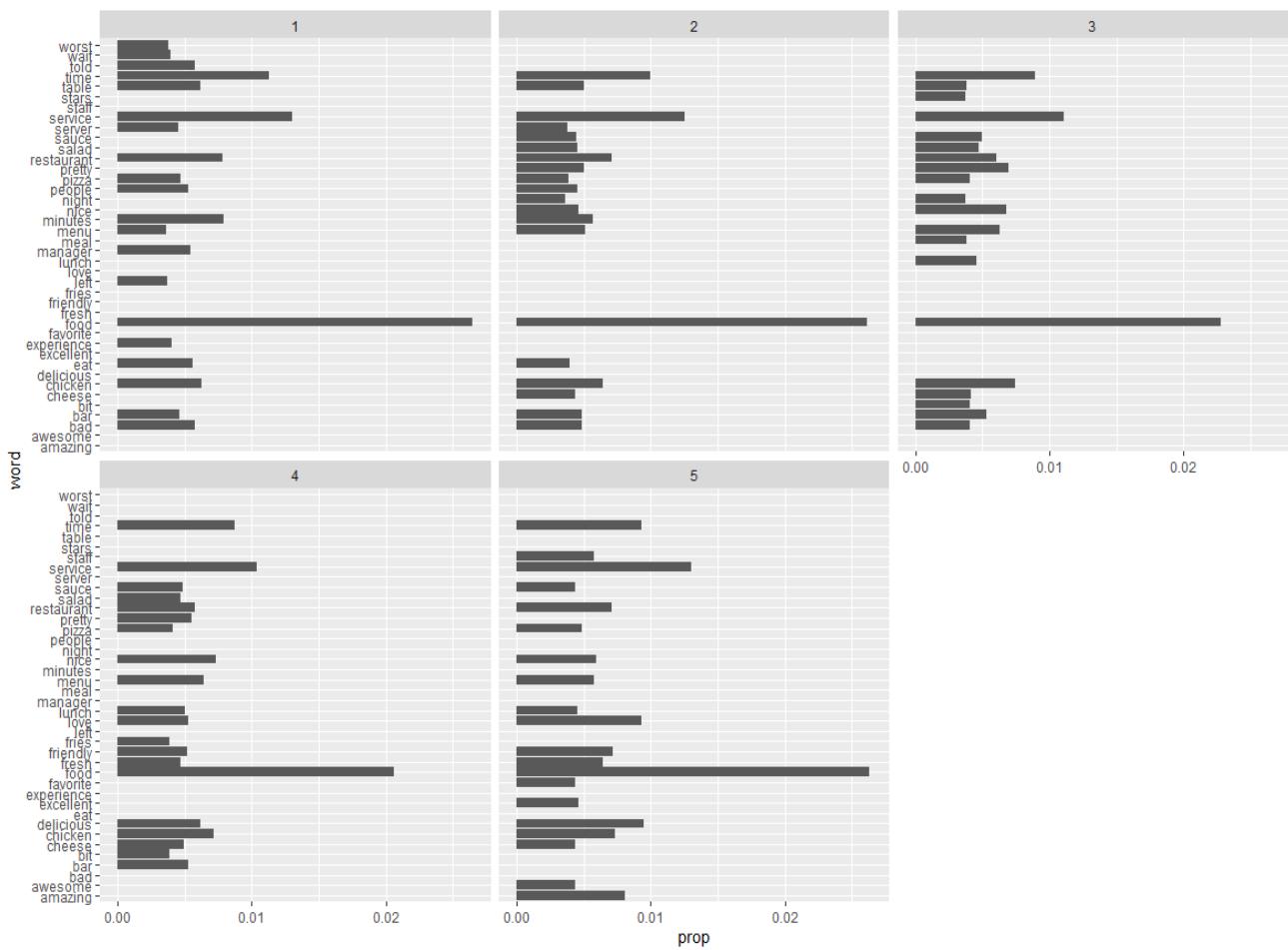


Tokenize data:

Grouping based on star rating and analyze proportion of various words and how they contribute to star ratings:

```
> # grouping based on star rating
> wordset <- rrdf %>% group_by(starsReview)
> # proportion for each word
> wordsetprop <- wordset %>% count(word, sort=TRUE) %>% mutate(prop=n/sum(n))
> wordsetprop %>% arrange(starsReview, desc(prop)) %>% filter(row_number(starsReview)<=20) %>% View()
> |
```

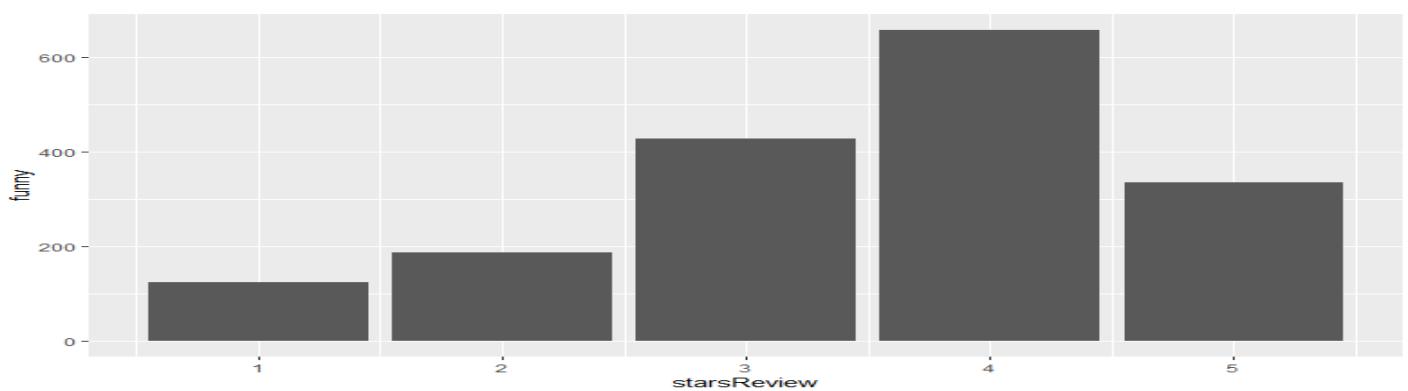
Plotting the graph for word against proportion for each rating:



We observed that words like worst, left, bad, wait, time appear a lot more than in ratings 1 and 2 while words like love, friendly, fresh, nice, awesome, amazing appear a lot more in ratings 4 and 5. This implies that various words can be used to assign labels to the review as ‘positive’ or ‘negative’; where ‘positive’ can be used for ratings 4 and 5 and ‘negative’ can be used for ratings 1 and 2.

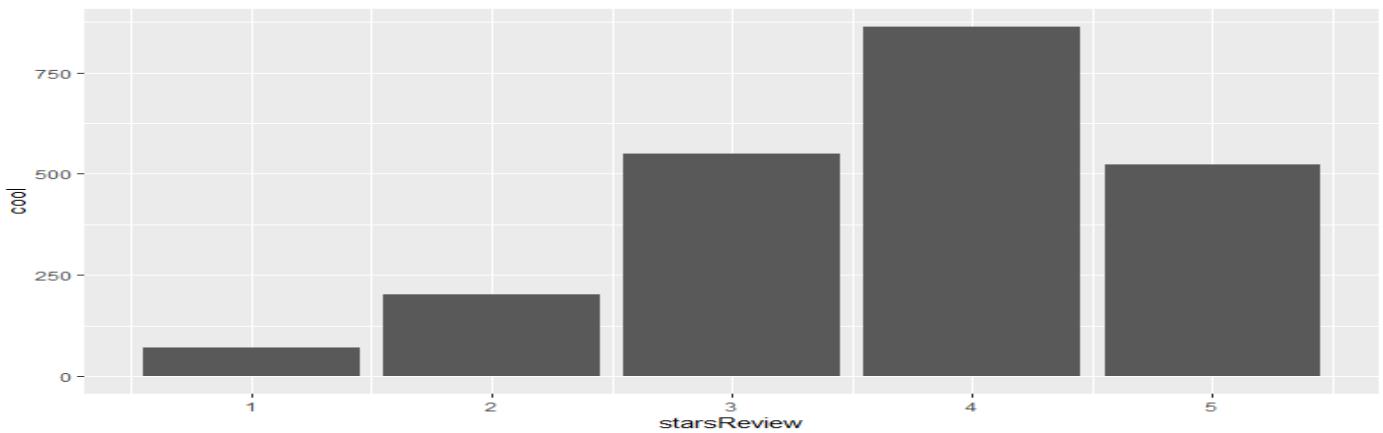
‘funny’ reviews:

```
> # finding relation to funny, cool and useful
> # FUNNY Reviews
> funnyReview <- wordset %>% select(starsReview, funny) %>% count(funny, sort=TRUE)
> # plot on graph
> funnyReview %>% arrange(starsReview, desc(funny)) %>% ggplot(aes(starsReview, funny))+geom_col()
```



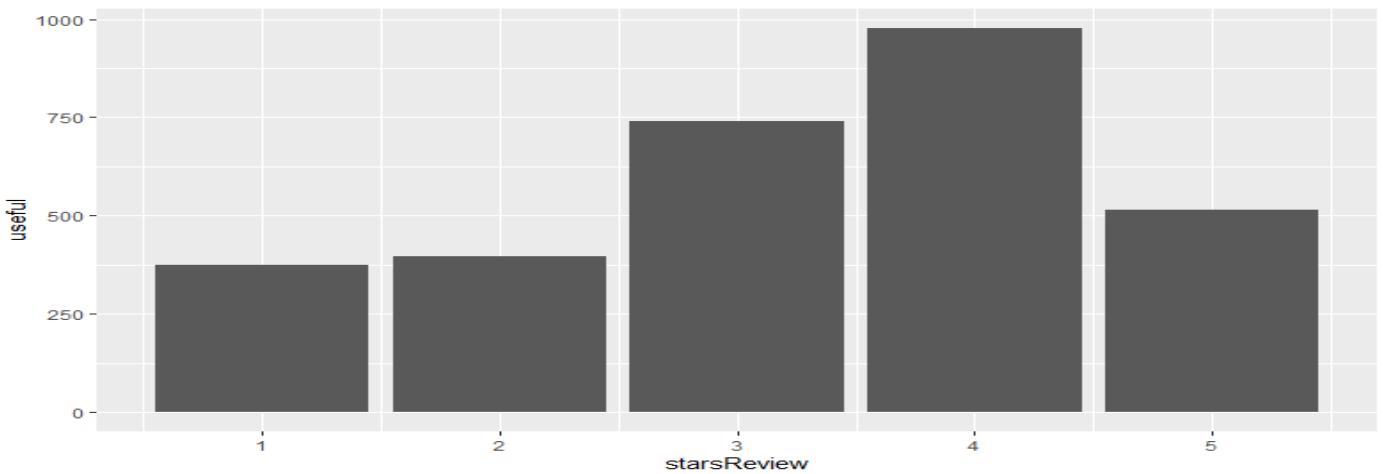
‘cool’ reviews:

```
> # COOL Reviews
> coolReview <- wordset %>% select(starsReview, cool) %>% count(cool, sort=TRUE)
> # plot on graph
> coolReview %>% arrange(starsReview, desc(cool)) %>% ggplot(aes(starsReview, cool))+geom_col()
```



'useful' reviews:

```
> # USEFUL Reviews
> usefulReview <- wordset %>% select(starsReview, useful) %>% count(useful, sort=TRUE)
> # plot on graph
> usefulReview %>% arrange(starsReview, desc(useful)) %>% ggplot(aes(starsReview, useful))+geom_col()
```

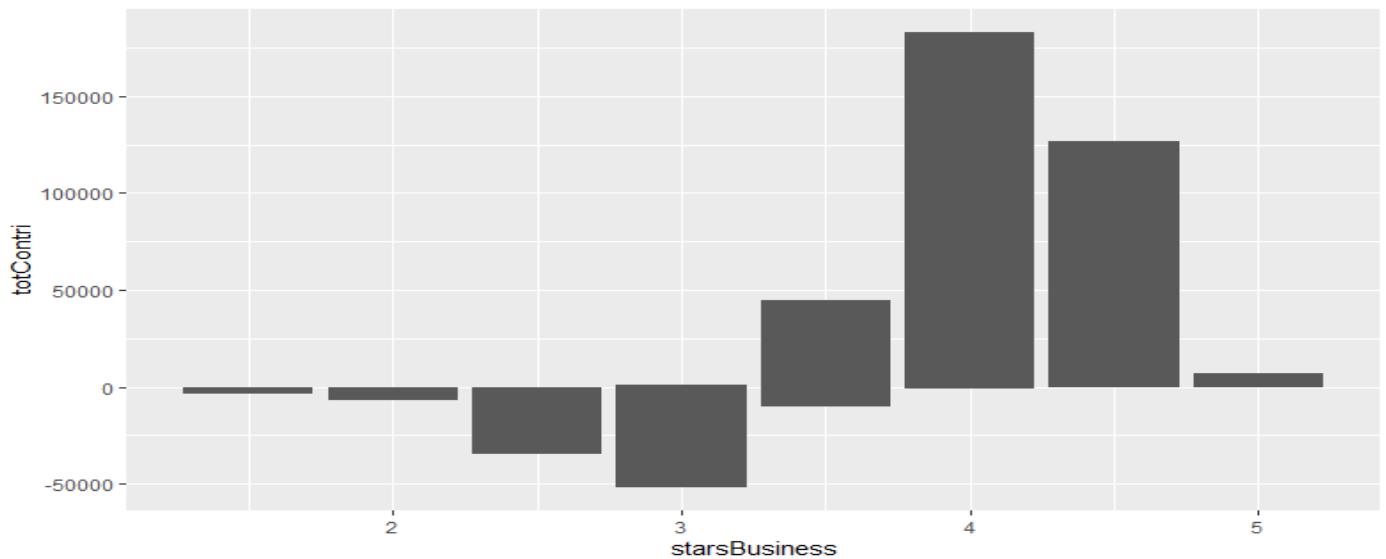


There is no strong relationship between ratings and tags like 'funny', 'cool', 'useful'. However, for tags funny and cool – majority of the users interacted with ratings 3, 4 and 5. But for 'useful' tags the interaction for star ratings 1 and 2 also have significant counts. Which implies that 'useful' tag helps users identify which restaurants they select. The above pattern does not completely emphasize the sentiment behind the review.

(ii) Manipulating star ratings to assess business star rating:

```
> ##### PART II #####
> busSet <- rrdf %>% group_by(business_id, starsBusiness)%>% count(starsReview) %>% mutate(contri=ifelse(starsReview<3.5, -1, 1), totCont
ri=sum(n*contri))
> # proportion of contribution towards business id
> busSetProp <- busSet %>% distinct(totContri)
> busSetProp %>% ungroup()
# A tibble: 457 x 3
   business_id      starsBusiness totContri
   <chr>                <dbl>      <dbl>
 1 -K3kqmykKlhlB4arCsLHOw        3     -588
 2 -lJtyCOTViNwUsU9YF120A       3.5      274
 3 -N_YCDH4HijYnJ-RisQfHA       3.5      201
 4 -OEIW0d096-492qa_luxaw       4      4993
 5 -sjCxkxv6xU5rEVLfybAuA      3.5      550
 6 -Ut87cwGFs03444Rd11p0Q      3.5      144
 7 -wtduWBW-U0XKCcGxz0twA       3     -558
 8 -YGePLsJ2pYccR3oaeCSAw      2.5     -100
 9 __zA29wBG0LleSxMzNHpwQ       4      990
10 _7u6Cdgo065xqUN0uRX4Ew       4      376
# ... with 447 more rows
> data <- busSetProp %>% arrange(starsBusiness, desc(totContri)) %>% View()
> busSetProp %>% arrange(starsBusiness, desc(totContri)) %>% ggplot(aes(starsBusiness, totContri))+geom_col()
```

Plotting graph for total contribution of various star ratings to a particular business id:



We can identify a clear trend that for business star ratings 1, 2 and 3 the contribution of low star ratings (1, 2 and 3) in the review is higher thereby making a negative impact on the rating. Whereas, for business star ratings 4 and 5, the contribution of high star ratings (like 4 and 5) in the review is higher and makes a positive impact. This impact is very visible in the business star ratings 3 and

3.5. A detailed analysis can be found in the data table.

Q2. 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).

Ans. Using occurrence as a measure to assign ‘positive’/ ‘negative’ sentiment to the review:

```
> ##### QUESTION 2 #####
> ##### pruning highest and lowest frequency of words #####
> wrds <- wordsetprop %>% group_by(word) %>% summarise( totWS= sum(starsReview*prop))
>
> ##### highest #####
> wrds %>% top_n(20)
>Selecting by totWS
# A tibble: 20 × 2
  word      totWS
  <chr>    <dbl>
1 bar      0.0697
2 cheese   0.0661
3 chicken  0.107 
4 delicious 0.0827
5 eat      0.0587
6 food     0.361 
7 fresh    0.0651
8 friendly 0.0729
9 love     0.0831
10 lunch   0.0652
11 menu    0.0870
12 nice    0.0915
13 pizza   0.0653
14 pretty  0.0661
15 restaurant 0.0986
16 salad   0.0669
17 sauce   0.0683
18 service 0.178 
19 staff   0.0629
20 time    0.139 

> ##### lowest #####
> wrds %>% top_n(-20)
>Selecting by totWS
# A tibble: 20 × 2
  word      totWS
  <chr>    <dbl>
1 apartment 0.000526
2 argue      0.000516
3 barcelona 0.000524
4 delay     0.000508
5 embarrassed 0.000487
6 excuses   0.000456
7 flies     0.000487
8 fucking   0.000447
9 grey      0.000521
10 handling 0.000525
11 huh       0.000524
12 inconvenience 0.000494
13 lousy    0.000476
14 pathetic 0.000512
15 practice 0.000513
16 presence 0.000506
17 responded 0.000459
18 stating   0.000516
19 unhappy   0.000520
20 wasting   0.000497
```

Taking a look at highest frequency words, it is immediate that words like – love, delicious, fresh, friendly, service, staff and many more, can be translated as compliment (positive). Example – ‘friendly service’, ‘delicious food’ and much more. And from the lowest frequent words, like – argue, delay, flies, lousy, unhappy, pathetic and many more which convey ‘dissatisfaction’ (‘negative’). Example – ‘lousy service’, ‘unhappy customer experience’, ‘flies’ (unhygienic) and a lot more.

Q3. 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 on the web. The first provides lists of words denoting different sentiment (for eg, positive, negative, joy, fear, anticipation, ...), 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.

How many matching terms are there for each of the dictionaries?

Consider using the dictionary based positive and negative terms to predict sentiment (positive or negative based on star rating) of a movie. One approach for this is: 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. Describe how you obtain predictions based on aggregated scores. Are you able to predict review sentiment based on these aggregated scores, and how do they perform? Does any dictionary perform better?

Ans. A look at the dictionaries:

1. Bing: <https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>
2. NRC: <http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm>
3. AFINN: [http://www2.imm.dtu.dk/pubdb/views/publication\\_details.php?id=6010](http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010)

One can check above dictionaries (used as reference in assignment) by navigating to the corresponding address. All three dictionaries analyze sentiment in a different way. Bing categorizes sentiment as ‘positive’ or ‘negative’. NRC measures each word under different sentiments (eg, joy, fear, positive, negative, ...). AFINN follows the concept of assigning ‘scores’ to words on a scale of -5 to +5. A brief look at the words and dictionary can be viewed below.

BING

	word	sentiment
1	2-faces	negative
2	abnormal	negative
3	abolish	negative
4	abominable	negative
5	abominably	negative
6	abominate	negative
7	abomination	negative
8	abort	negative
9	aborted	negative
10	aborts	negative
11	abound	positive
12	abounds	positive
13	abrade	negative
14	abrasive	negative
15	abrupt	negative
16	abruptly	negative
17	abscond	negative
18	absence	negative
19	absent-minded	negative
20	absentee	negative
21	absurd	negative
22	absurdity	negative
23	absurdly	negative
24	absurdness	negative
25	abundance	positive
26	abundant	positive
27	abuse	negative
28	abused	negative
29	abuses	negative
30	abusive	negative
31	abyssmal	negative
32	abysmally	negative
33	abyss	negative
34	accessible	positive
35	accessible	positive
36	accidental	negative
37	accidem	positive

NRC

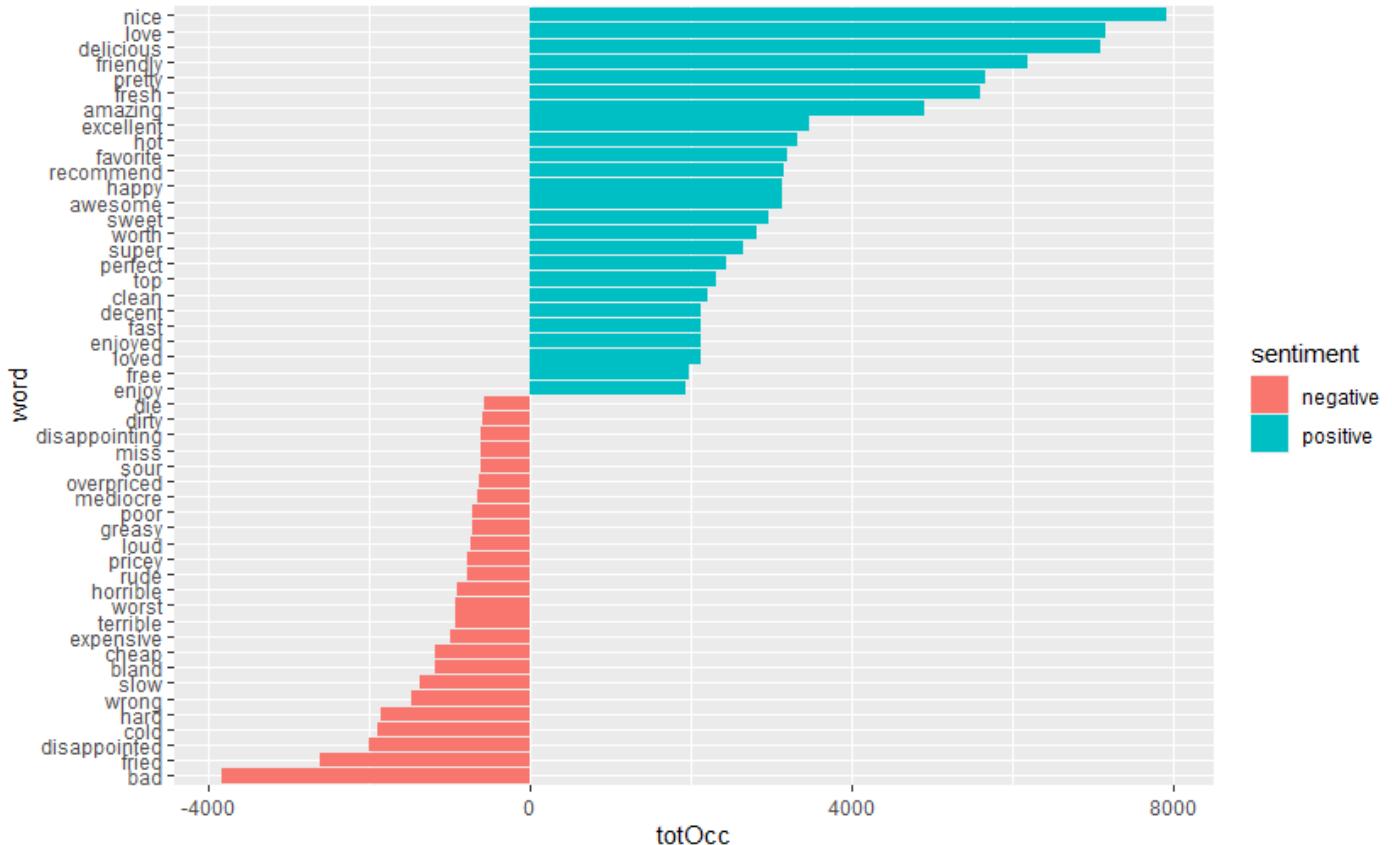
	word	sentiment
1	abacus	trust
2	abandon	fear
3	abandon	negative
4	abandon	sadness
5	abandoned	anger
6	abandoned	fear
7	abandoned	negative
8	abandoned	sadness
9	abandonment	anger
10	abandonment	fear
11	abandonment	negative
12	abandonment	sadness
13	abandonment	surprise
14	abba	positive
15	abbot	trust
16	abduction	fear
17	abduction	negative
18	abduction	sadness
19	abduction	surprise
20	aberrant	negative
21	aberration	disgust
22	aberration	negative
23	abhor	anger
24	abhor	disgust
25	abhor	fear
26	abhor	negative
27	abhorrent	anger
28	abhorrent	disgust
29	abhorrent	fear
30	abhorrent	negative
31	ability	positive
32	abject	disgust
33	abject	negative
34	abnormal	disgust
35	abnormal	negative
36	abolish	anger

AFINN

	word	value
1	abandon	-2
2	abandoned	-2
3	abandons	-2
4	abducted	-2
5	abduction	-2
6	abductions	-2
7	abhor	-3
8	abhorred	-3
9	abhorrent	-3
10	abhors	-3
11	abilities	2
12	ability	2
13	aboard	1
14	absentee	-1
15	absentees	-1
16	absolve	2
17	absolved	2
18	absolves	2
19	absolving	2
20	absorbed	1
21	abuse	-3
22	abused	-3
23	abuses	-3
24	abusive	-3
25	accept	1
26	accepted	1
27	accepting	1
28	accepts	1
29	accident	-2
30	accidental	-2
31	accidentally	-2
32	accidents	-2
33	accomplish	2
34	accomplished	2
35	accomplishes	2
36	accusation	-2

**'Bing' dictionary:** Computing the total occurrence of various words, comparing them to the dictionary (assigning sentiment) and modifying total occurrence based on the sentiments ('positive': +(total occurrence), 'negative': -(total occurrence)). On observing the top most occurring words, the positive sentiment words appear the most in reviews and observing the lowest occurring words are assigned negative sentiment. We can plot the trend as well.

```
> ##### BING #####
> get_sentiments("bing") %>% View()
> rrSenti_bing <- rrTokens %>% inner_join(get_sentiments("bing"), by="word")
> rrSenti_bingOcc <- rrSenti_bing %>% group_by(word, sentiment) %>% count(sentiment) %>% summarise(totOcc=sum(n)) %>% arrange(sentiment,
  desc(totOcc))
`summarise()` has grouped output by 'word'. You can override using the `groups` argument.
>
> # negate the counts for the negative sentiment words
> rrSenti_bingOcc <- rrSenti_bingOcc %>% mutate(totOcc=ifelse(sentiment=="positive", totOcc, -totOcc))
>
> # which are the most positive and most negative words in reviews
> rrSenti_bingOcc <- ungroup(rrSenti_bingOcc)
>
> rrSenti_bingOcc %>% top_n(25)
Selecting by totOcc
# A tibble: 25 x 3
  word      sentiment totOcc
  <chr>    <chr>     <int>
1 nice     positive   7907
2 love     positive   7145
3 delicious positive  7089
4 friendly positive  6180
5 pretty   positive  5671
6 fresh    positive  5602
7 amazing   positive  4906
8 excellent positive 3484
9 hot      positive  3321
10 favorite positive 3201
# ... with 15 more rows
> rrSenti_bingOcc %>% top_n(-25)
Selecting by totOcc
# A tibble: 25 x 3
  word      sentiment totOcc
  <chr>    <chr>     <int>
1 bad      negative -2827
2 fried    negative -2665
3 disappointed negative -2004
4 cold     negative -1896
5 hard     negative -1852
6 wrong    negative -1460
7 slow     negative -1358
8 bland    negative -1184
9 cheap    negative -1179
10 expensive negative -980
# ... with 15 more rows
>
> # plot them on graph
> rbind(top_n(rrSenti_bingOcc, 25), top_n(rrSenti_bingOcc, -25)) %>% mutate(word=reorder(word,totOcc)) %>% ggplot(aes(word, totOcc, fill=sentiment)) +geom_col() +coord_flip()
Selecting by totOcc
Selecting by totOcc
```

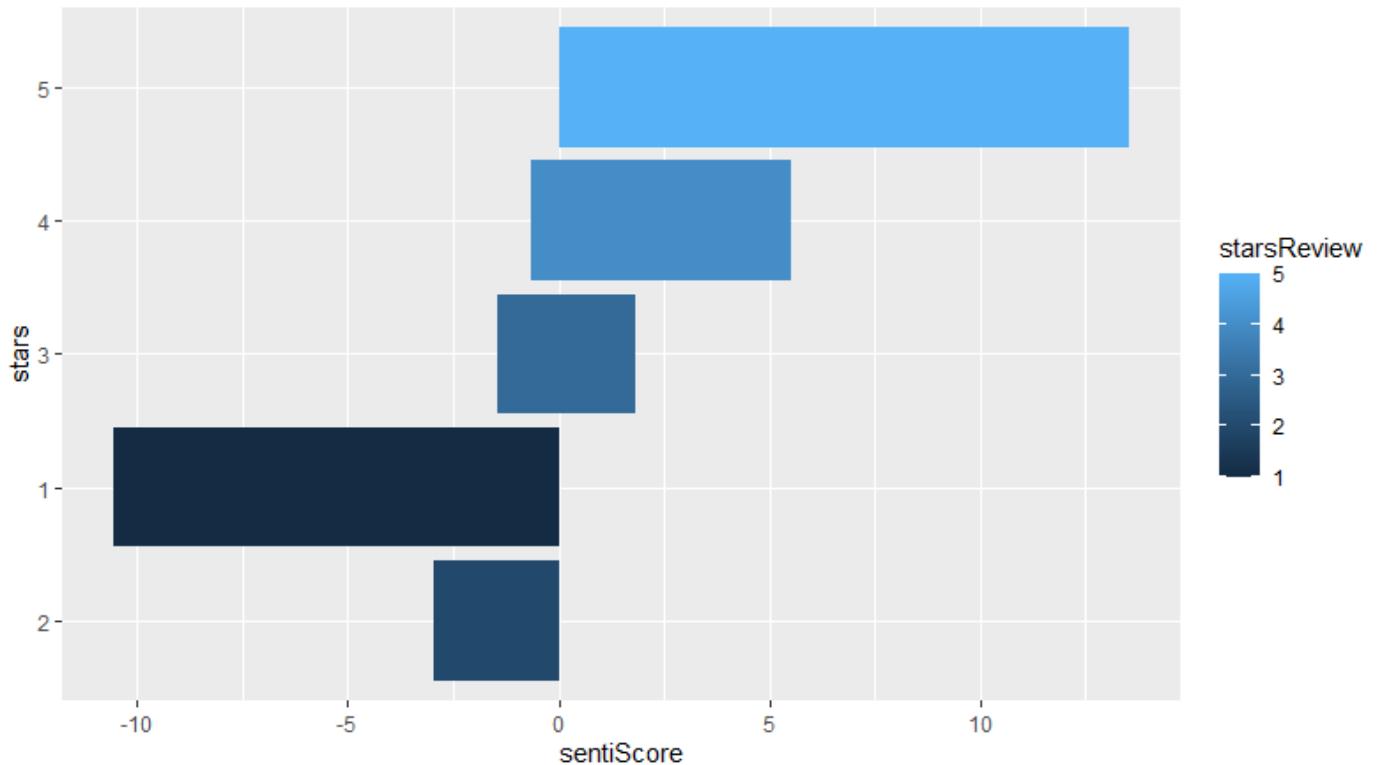


Using a similar approach we can analyze sentiments for reviews:

```

> # sentiment for reviews
> # positive/negative sentiment words per review
> rvSenti_bing <- rrSenti_bing %>% group_by(review_id, starsReview) %>% summarise(nwords=n(), posSum=sum(sentiment=='positive'), negSum=sum(sentiment=='negative'))
`summarise()` has grouped output by 'review_id'. You can override using the `.`groups` argument.
>
> # calculate sentiment score based on proportion of positive, negative words
> rvSenti_bing <- rvSenti_bing %>% mutate(posProp=posSum/nwords, negProp=negSum/nwords)
>
> rvSenti_bing <- rvSenti_bing %>% mutate(sentiScore=posProp-negProp)
>
> temp <- rvSenti_bing %>% filter(nwords > 20) %>% arrange(nwords, desc(sentiScore))
>
> temp <- ungroup(temp)
>
> temp %>% top_n(20) %>% View()
Selecting by sentiScore
>
> temp %>% top_n(-20) %>% View()
Selecting by sentiScore
>
> # plot them on graph
> rbind(top_n(temp, 20), top_n(temp, -20)) %>% mutate(stars=reorder(starsReview, sentiScore)) %>% ggplot(aes(stars, sentiScore, fill=starsReview))
Selecting by sentiScore
Selecting by sentiScore
> |

```



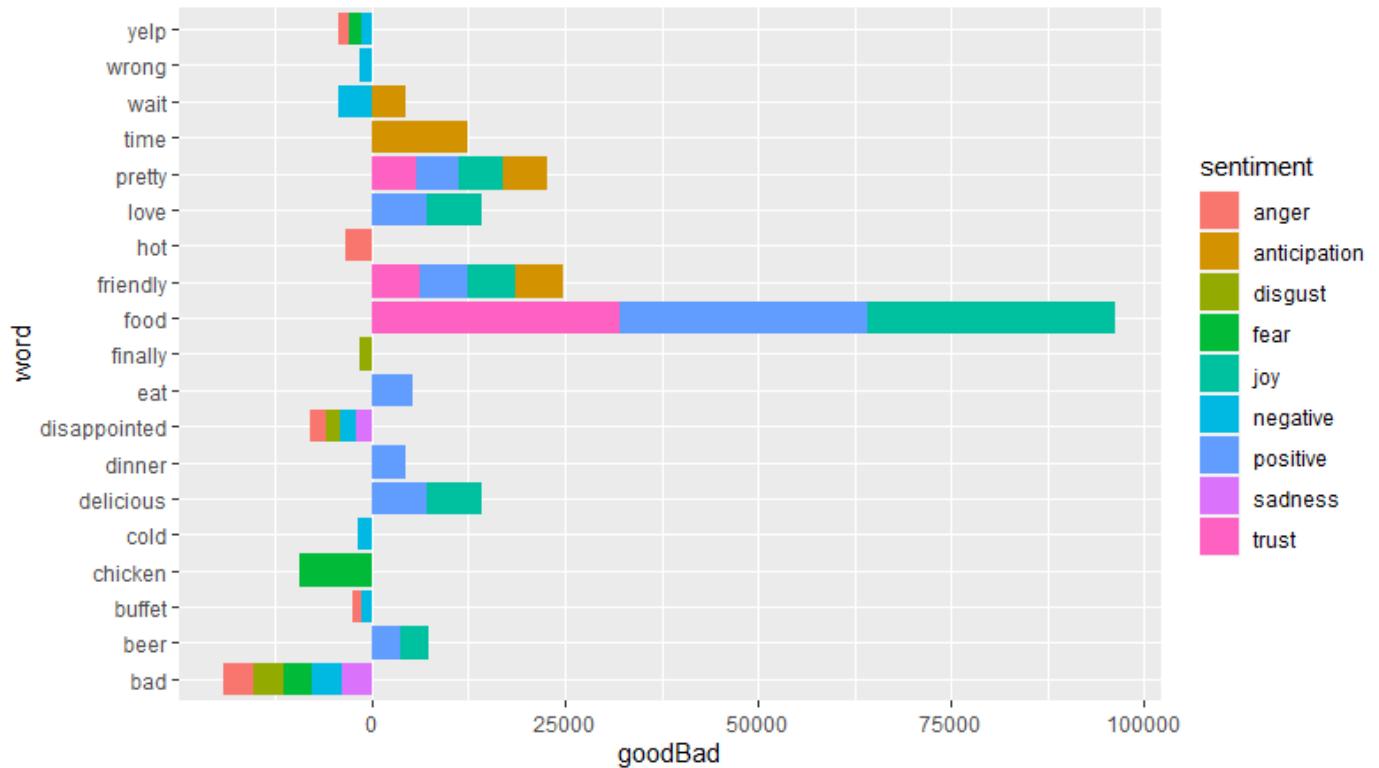
**NRC' dictionary:** We tried a similar approach of counting the occurrences but when modifying the occurrence to reflect positive and negative words, we used various sentiments mentioned in the dictionary to group words as good or bad. Words belonging to groups like ‘joy’, ‘positive’, ‘anticipation’ and more, were ‘positive sentiment’ and positive total occurrence was assigned to them whereas words like ‘anger’, ‘disgust’, ‘fear’ and more were ‘negative sentiment’ and negative total occurrence was assigned. A look at the selected 20 rows from top depict a higher occurrence for sentiments like - ‘joy’, ‘anticipation’ whereas the rows from bottom have higher occurrences for sentiments like - ‘disgust’, ‘anger’. We can plot the trend for a better understanding.

```

> get_sentiments("nrc") %>% View()
>
> rrSenti_nrc <- rrTokens %>% inner_join(get_sentiments("nrc"), by="word")
>
> rrSenti_nrc0cc <- rrSenti_nrc %>% group_by(word, sentiment) %>% count(sentiment) %>% summarise(totOcc=sum(n)) %>% arrange(sentiment, d
esc(totOcc))
`summarise()` has grouped output by 'word'. You can override using the ``.groups` argument.
>
> #How many words are there for the different sentiment categories
> rrSenti_nrc0cc %>% group_by(sentiment) %>% summarise(count=n(), sumn=sum(totOcc))
# A tibble: 10 x 3
  sentiment   count   sumn
  <chr>     <int>  <int>
1 anger      749   37299
2 anticipation 617   92916
3 disgust     660   30484
4 fear        855   39280
5 joy         537   138528
6 negative    2016   85460
7 positive    1641  239922
8 sadness     730   37560
9 surprise     381   36875
10 trust      822   131379
>
> #top few words for different sentiments
> rrSenti_nrc0cc%>% group_by(sentiment) %>% top_n(10) %>% View()
Selecting by totOcc
>
> rrSenti_nrc0cc <- rrSenti_nrc0cc %>% mutate(goodBad=ifelse(sentiment %in% c('anger', 'disgust', 'fear', 'sadness', 'negative'), -totO
cc, ifelse(sentiment %in% c('positive', 'joy', 'anticipation', 'trust', 'surprise'), totOcc, 0)))
>
> rrSenti_nrc0cc <- ungroup(rrSenti_nrc0cc)
>
```

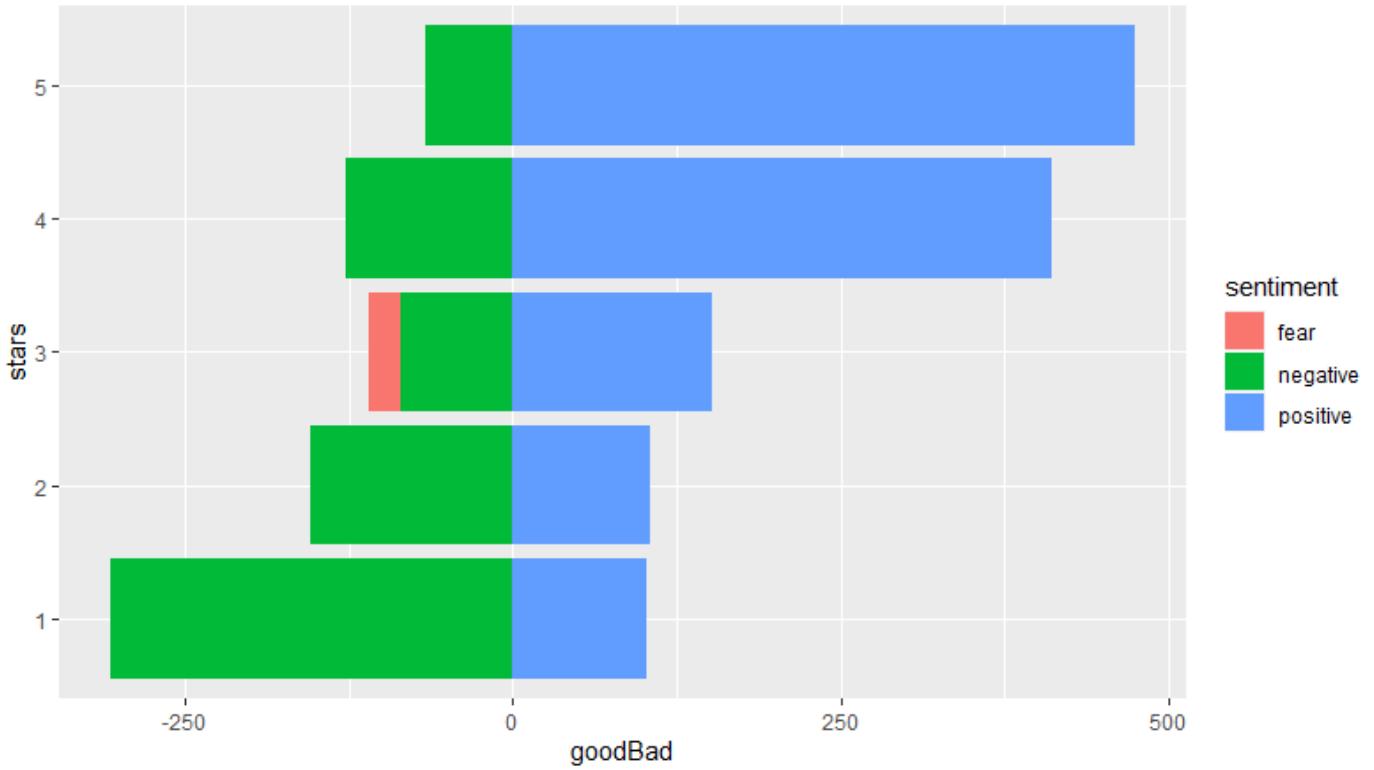
```

> top_n(rrSenti_nrc0cc, 20)
Selecting by goodBad
# A tibble: 21 x 4
  word      sentiment   totOcc  goodBad
  <chr>     <chr>     <int>   <dbl>
1 time      anticipation 12520   12520
2 friendly  anticipation  6180   6180
3 pretty    anticipation  5671   5671
4 wait      anticipation  4307   4307
5 food      joy          32111  32111
6 love      joy          7145   7145
7 delicious joy          7089   7089
8 friendly  joy          6180   6180
9 pretty    joy          5671   5671
10 beer     joy          3693   3693
# ... with 11 more rows
> top_n(rrSenti_nrc0cc, -20)
Selecting by goodBad
# A tibble: 20 x 4
  word      sentiment totOcc  goodBad
  <chr>     <chr>     <int>   <dbl>
1 bad       anger      3827   -3827
2 hot       anger      3321   -3321
3 disappointed  anger  2004   -2004
4 yelp      anger      1437   -1437
5 buffet    anger      1253   -1253
6 bad       disgust    3827   -3827
7 disappointed disgust  2004   -2004
8 finally   disgust    1576   -1576
9 chicken   fear       9411   -9411
10 bad      fear       3827   -3827
```



Similarly, we observed the pattern in the reviews:

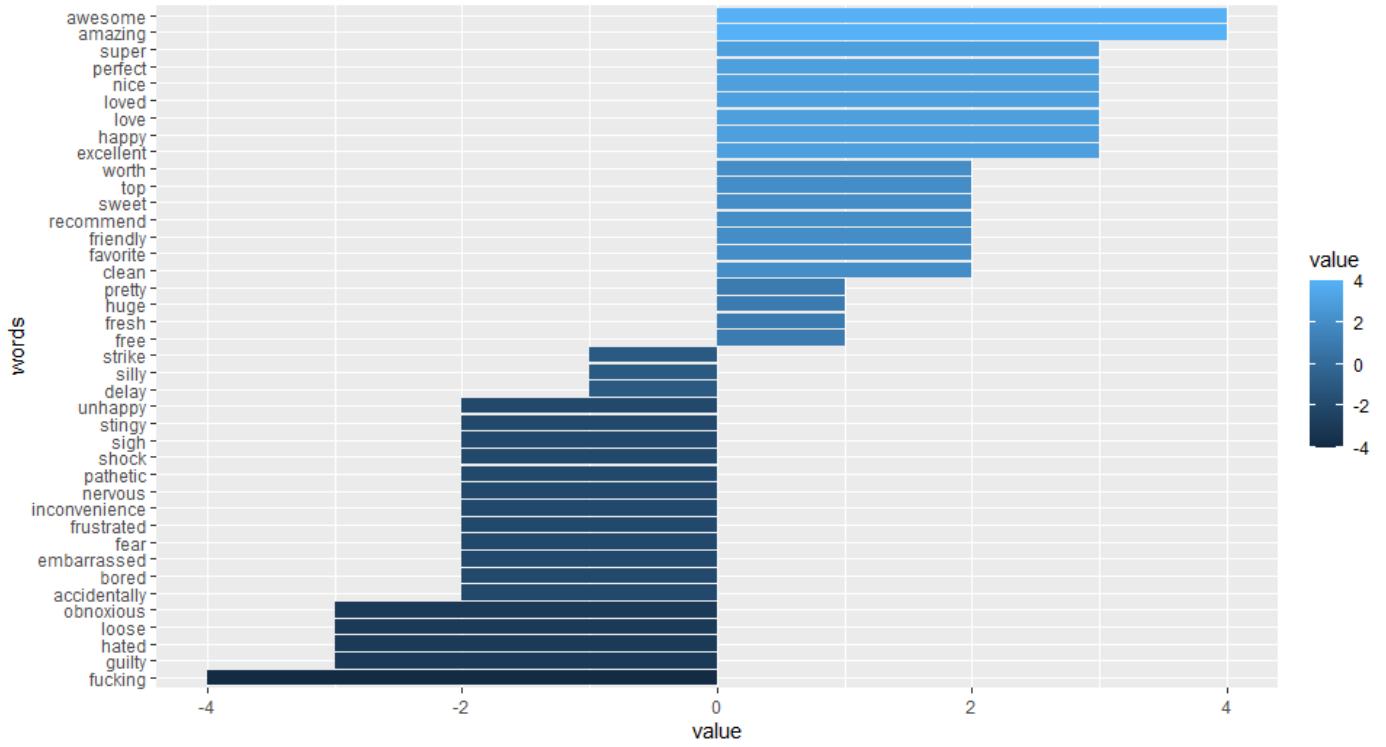
```
> # sentiment for reviews
> # positive/negative sentiment words per review
> rvSenti_nrc <- rrSenti_nrc %>% group_by(review_id, starsReview, sentiment) %>% count(sentiment) %>% summarise(totOcc=sum(n)) %>% mutate(goodBad=ifelse(sentiment %in% c('anger', 'disgust', 'fear', 'sadness', 'negative'), -totOcc, ifelse(sentiment %in% c('positive', 'joy', 'anticipation', 'trust', 'surprise'), totOcc, 0))) %>% arrange(sentiment, desc(totOcc))
`summarise()` has grouped output by `review_id`, `starsReview`. You can override using the `.`groups` argument.
>
> rvSenti_nrc <- ungroup(rvSenti_nrc)
>
> top_n(rvSenti_nrc, 20)
Selecting by goodBad
# A tibble: 23 x 5
  review_id      starsReview sentiment totOcc goodBad
  <chr>          <int>    <chr>     <int>   <dbl>
1 N3ysL-pleicEnvx6pAKNA      5 positive    74     74
2 LEYmZHxXnbz2qfxj6GyRQ      5 positive    68     68
3 viSm0GMD0oD2MrbQy_mFiA     5 positive    61     61
4 yZ2c0TXVVDjEAkK-gZsfFGQ    5 positive    59     59
5 0B8RfcLGLkjw35JnfWRRCa    4 positive    57     57
6 eKsXeF5JgdmSBaoU0mARJQ     5 positive    57     57
7 3bKvxeQx9r5f7znEjyBD2w    5 positive    54     54
8 GkpvtZM8KK7Lfj4cRe52g      2 positive    54     54
9 SAC2c5wt7I-gx80tY9zWQ      4 positive    54     54
10 QQCi2J7ESbQ46n0QTToA_Eg   3 positive    53     53
# ... with 13 more rows
> top_n(rvSenti_nrc, -20)
Selecting by goodBad
# A tibble: 31 x 5
  review_id      starsReview sentiment totOcc goodBad
  <chr>          <int>    <chr>     <int>   <dbl>
1 Pp7_XQLWQJOZ5UGo5YbrKg    3 fear       25    -25
2 AE3TMM9uCHw711IOSHr79A     2 negative   37    -37
3 Pp7_XQLWQJOZ5UGo5YbrKg    3 negative   37    -37
4 XCrmFge99svExxukBVejkg   4 negative   32    -32
5 HdWx9YpxpxiKgc1kM1PGw     1 negative   30    -30
6 CtpSXzoH4wGJlxk69x44g     1 negative   29    -29
7 3yUCBLmpHtBUaN5coiXZLA    2 negative   27    -27
8 S5dr9WL_kpIBi6GXD7p5LA    4 negative   27    -27
9 MDY_WjtTuZ3YN-6ZVCGXuQ    3 negative   26    -26
10 BvT6xd-cBmUIKi0WedXUjQ   1 negative   25    -25
# ... with 21 more rows
>
> # plot them on graph
> rbind(top_n(rvSenti_nrc, 20), top_n(rvSenti_nrc, -20)) %>% mutate(stars=reorder(starsReview, goodBad)) %>% ggplot(aes(stars, goodBad, fill=sentiment)) +geom_col() +coord_flip()
Selecting by goodBad
Selecting by goodBad
> |
```



On analyzing the pattern observed by plotting star ratings against total occurrence for various sentiments, we find that for star ratings 1 and 2, a higher proportion is attributed to negative sentiment, whereas for ratings 4 and 5 the trend is more positive. However, rating 3 has almost equal influence on both sentiments.

**'AFINN' dictionary:** With AFINN the approach was changed a bit, here we did analyze the total occurrence, but the total occurrence was based on the score/value assigned to the word by AFINN. So we filtered the highest and lowest frequent words based on the range of value and total occurrence. A look at the graph can be seen below.

```
> get_sentiments("afinn") %>% View()
>
> rrSenti_afinn<- rrTokens%>% inner_join(get_sentiments("afinn"), by="word")
>
> rrSenti_afinnOcc <- rrSenti_afinn %>% group_by(word, value) %>% count(value) %>% summarise(tot0cc=sum(n)) %>% arrange(value, desc(tot0cc))
`summarise()` has grouped output by 'word'. You can override using the `.`groups` argument.
>
> rrSenti_afinnOcc <- ungroup(rrSenti_afinnOcc)
>
> tempTop <- rrSenti_afinnOcc %>% filter(tot0cc > 50 & value > 0) %>% top_n(20)
Selecting by tot0cc
> tempBtm <- rrSenti_afinnOcc %>% filter(tot0cc > 50 & value <0) %>% top_n(-20)
Selecting by tot0cc
>
> # plot them on graph
> rbind(tempTop, tempBtm) %>% mutate(words=reorder(word, value)) %>% ggplot(aes(words, value, fill=value)) +geom_col() +coord_flip()
> View(rrSenti_afinnOcc)
> |
```

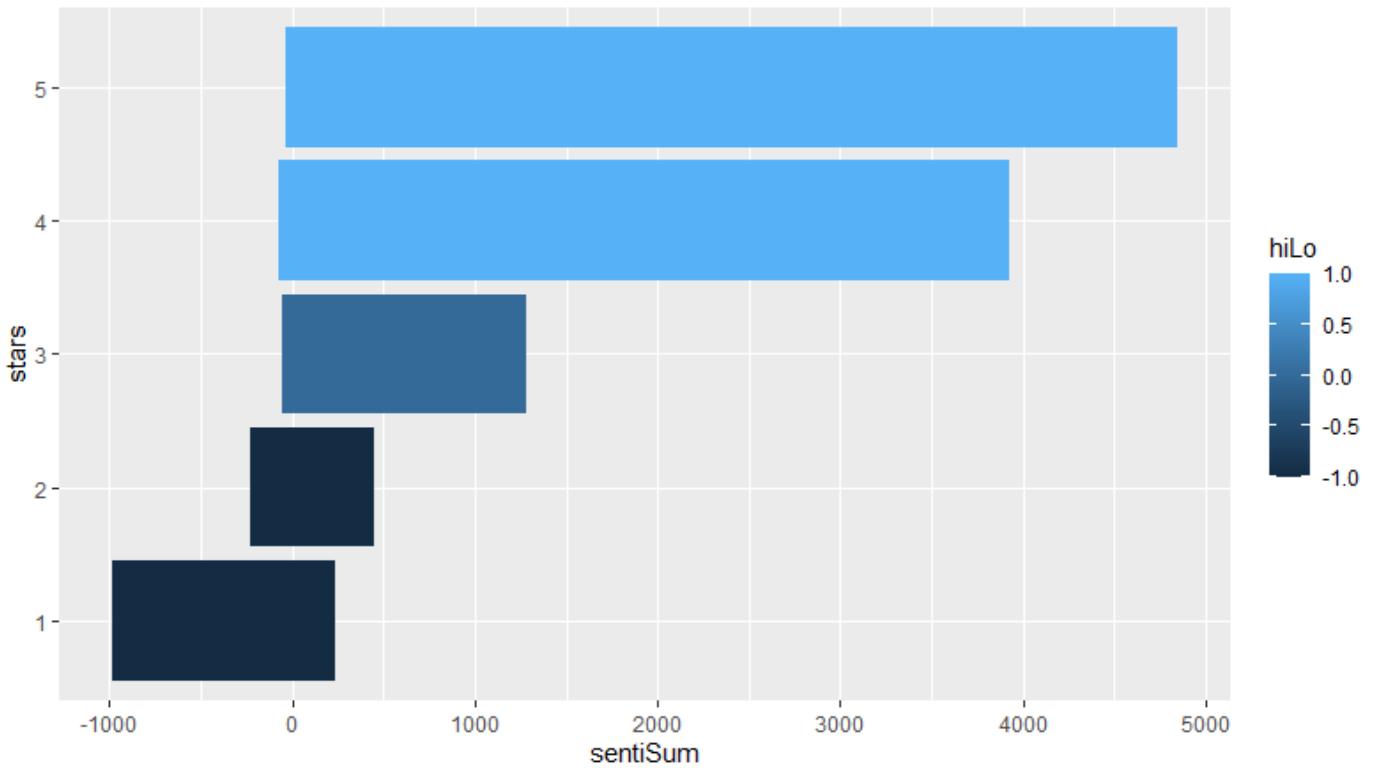


On careful analysis, we can observe that words like ‘awesome’, ‘amazing’, ‘excellent’, ‘happy’ are ranked on the +1 to +5 scale denoting a ‘positive’ sentiment while words like ‘delay’, ‘unhappy’, ‘embarrassed’, ‘unhappy’ are ranked on -1 to -5 scale denoting ‘negative’ sentiment.

Similarly, for reviews, we can conclude that:

```
> # sentiment for reviews
> # positive/negative sentiment words per review
> rvSenti_afinn <- rrSenti_afinn %>% group_by(review_id, starsReview) %>% summarise(nwords=n(), sentiSum=sum(value)) %>% filter(nwords > 20) %>% arrange(nwords, desc(sentiSum))
`summarise()` has grouped output by 'review_id'. You can override using the `.`groups` argument.
>
> rvSenti_afinn %>% group_by(starsReview) %>% summarise(avgLen=mean(nwords), avgSenti=mean(sentiSum))
# A tibble: 5 x 3
  starsReview avgLen avgSenti
    <int>    <dbl>    <dbl>
1       1     25.2   -0.78
2       2     27.9    4.57
3       3     25.8    20.9 
4       4     26.9    31.3 
5       5     26.9    39.4 

> #considering reviews with 1 to 2 stars as negative, and this with 4 to 5 stars as positive
> rvSenti_afinn <- rvSenti_afinn %>% mutate(hiLo= ifelse(starsReview <= 2, -1, ifelse(starsReview >=4, 1, 0 )))
> rvSenti_afinn <- rvSenti_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
> afinnCal <- rvSenti_afinn %>% filter(hiLo!=0)
> table(actual=afinnCal$hiLo, predicted=afinnCal$pred_hiLo)
   predicted
actual -1  1
  -1  74 60
   1   9 236
>
> rvSenti_afinn <- ungroup(rvSenti_afinn)
>
> tempTop_afinn <- rvSenti_afinn %>% filter(nwords > 20) %>% top_n(20)
Selecting by pred_hiLo
> tempBtm_afinn <- rvSenti_afinn %>% filter(nwords > 20) %>% top_n(-20)
Selecting by pred_hiLo
>
> # plot them on graph
> rbind(tempTop_afinn, tempBtm_afinn) %>% mutate(stars=reorder(starsReview, sentiSum)) %>% ggplot(aes(stars, sentiSum, fill=hiLo)) +geom_col() +coord_flip()
> |
```



We observe that the distribution of the sentiment values have negative sentiments depicted by the negative integers which are visible for the star ratings 1 and 2. Star rating 3 shows a moderate distribution which implies the values lie somewhere around 0. But for ratings 4 and 5 it is evident that the majority of the value lies on the positive range.

Q4. 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). One may seek a model built using only the terms matching any or all of the sentiment dictionaries, or 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, svm, random forest (ranger).

(i) 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. Do you use term frequency, tfidf, or other measures, and why? What is the size of the documentterm matrix? Should you use stemming or lemmatization when using the dictionaries?

(ii) 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 here? Report on performance of the models. Compare performance with that in part (c) above. How do you evaluate performance? Which performance measures do you use, why.

Ans. i) We will first look at the performance of a single model amongst each dictionary and choose the best model amongst the four dictionaries used (Bing, AFINN, NRC, Combination of the three dictionaries). Thereafter, we will compare the best performing model of each dictionary with other dictionaries to conclude with the best performing model amongst all the models and libraries.

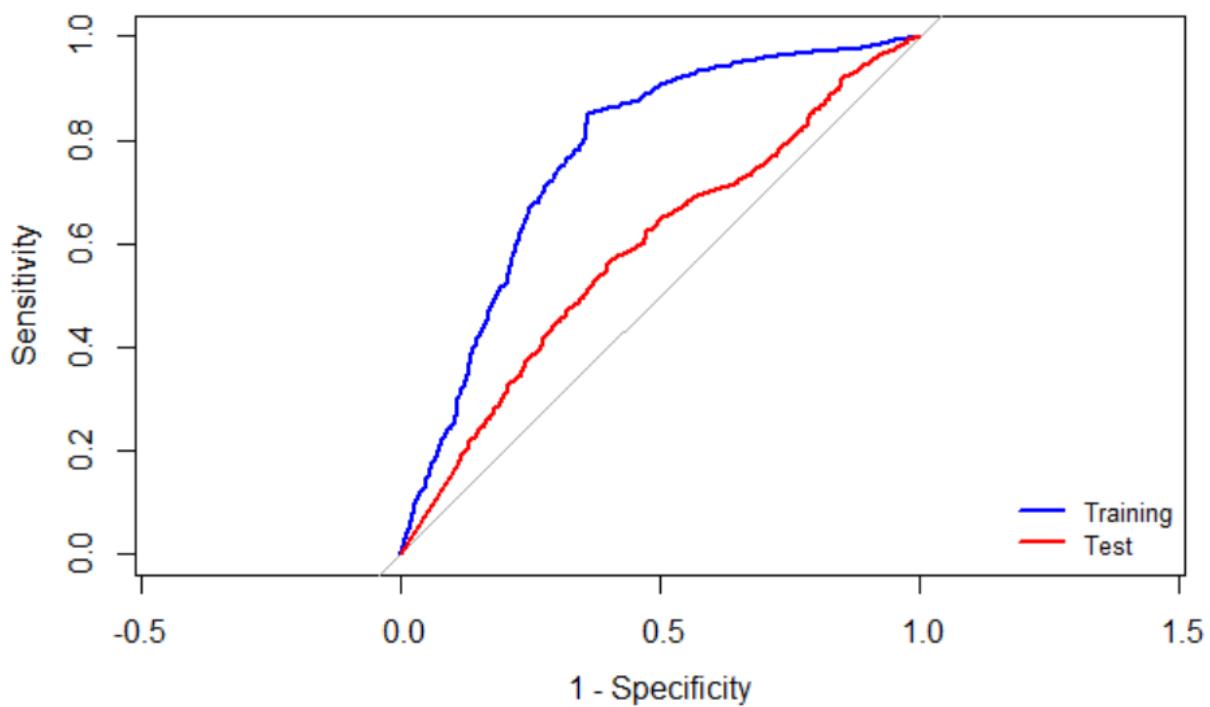
A) **Naïve bayes:**

- 1) **Bing dictionary**- On using naive bayes model on the dataset we observe the=at the training data has an AUC of 0.7652 and for test data the values is 0.5872, which implies that the combination of library and model is poorly fitted to the dataset.

```

predicted
actual FALSE TRUE
 -1 1603 1
  1 4688 5
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7652
Setting direction: controls < cases
predicted
actual FALSE TRUE
 -1 497 74
  1 1209 320
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.5872
Setting direction: controls < cases

```

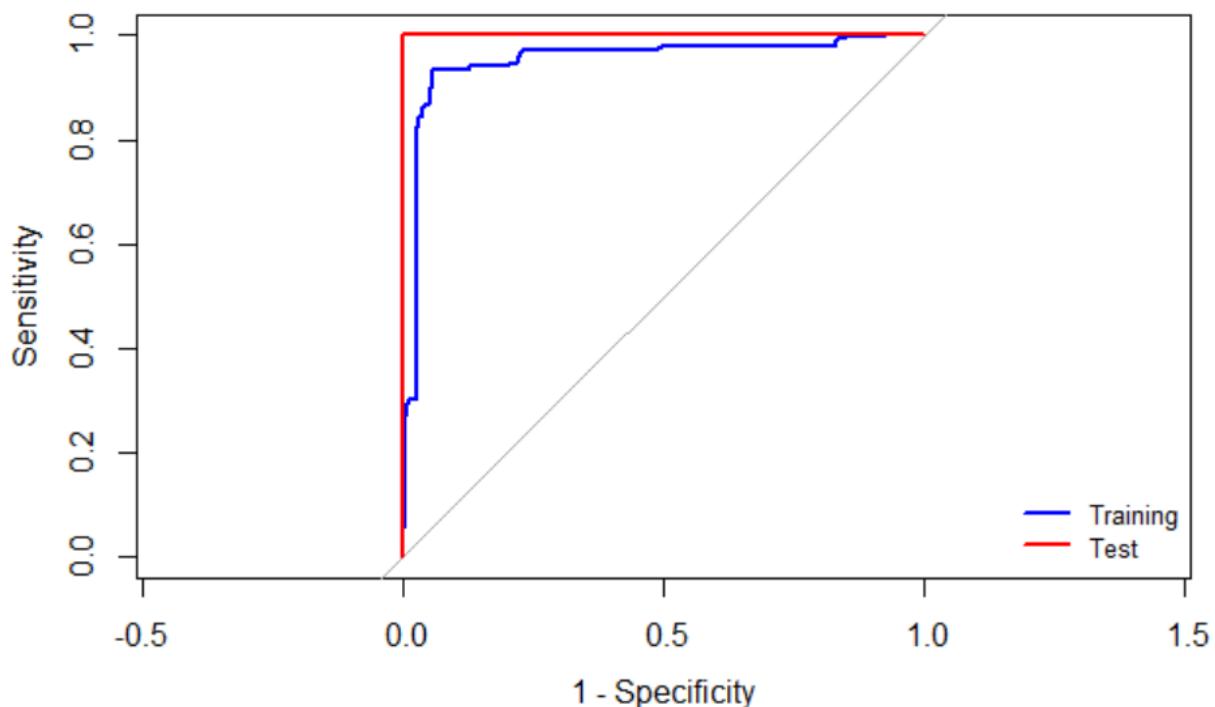


- 2) NRC dictionary- On using naive bayes model on the dataset we observe that the training data has an AUC of 0.9615 and for test data the values is 0.9989, which implies that the combination of library and model is overly fitted to the dataset.

```

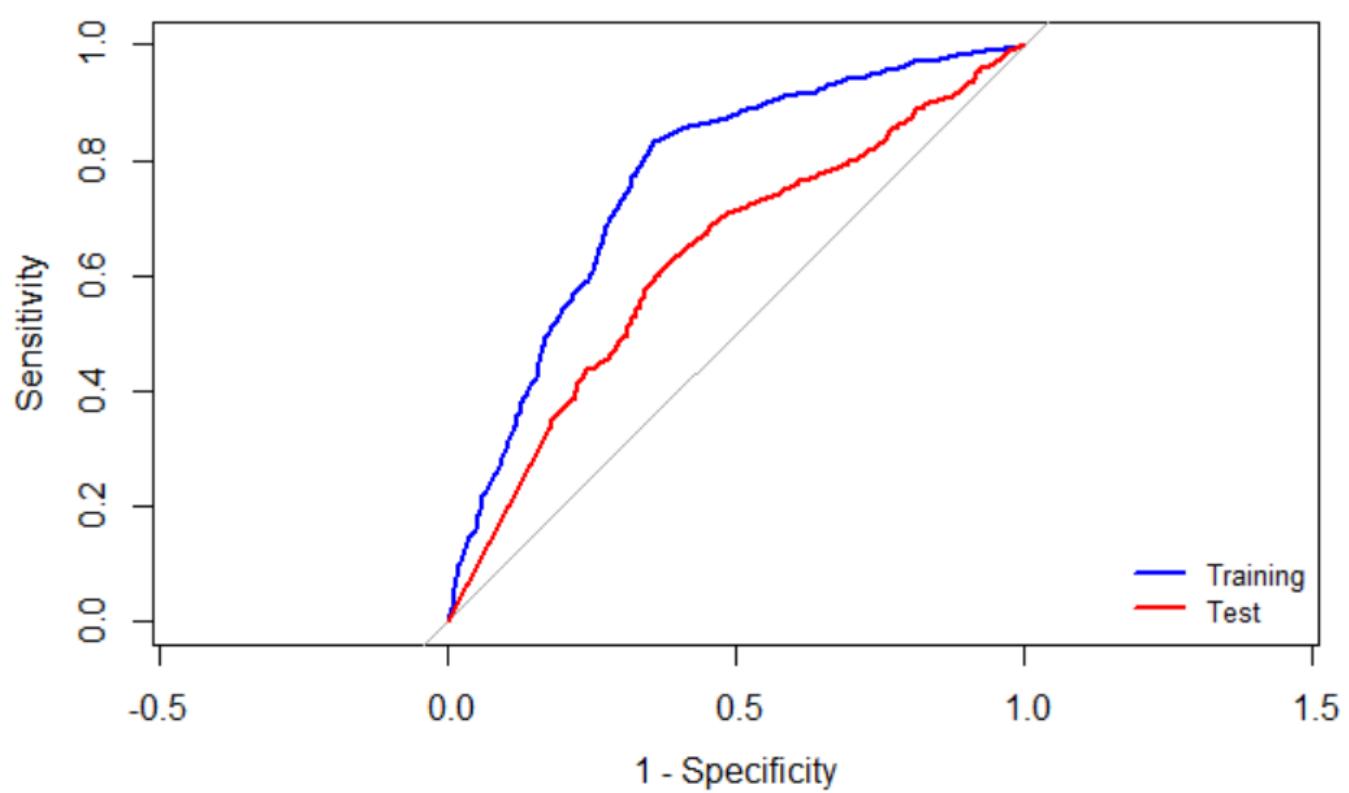
      predicted
actual FALSE TRUE
 -1 2092 101
  1   330 5094
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.9615
Setting direction: controls < cases
      predicted
actual FALSE TRUE
 -1    731   0
  1      4 1805
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.9989
Setting direction: controls < cases

```



- 3) [AFINN dictionary](#) -On using naive bayes model on the dataset we observe the=at the training data has an AUC of 0.7636 and for test data the values is 0.6269, which implies that the combination of library and model is poorly fitted to the dataset.

```
predicted
actual FALSE
-1 1633
 1 4652
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7636
Setting direction: controls < cases
predicted
actual FALSE TRUE
-1   386   162
 1   798   749
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6269
Setting direction: controls < cases
```



- 4) Combined dictionary - On using naive bayes model on the dataset we observe that the training data has an AUC of 0.6825 and for test data the value is 0.5569, which implies that the combination of library and model is poorly fitted to the dataset.

```

predicted
actual FALSE TRUE
-1 108 1991
1 55 5170
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6825
Setting direction: controls < cases
predicted
actual FALSE TRUE
-1 648 57
1 1555 182
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.5569
Setting direction: controls < cases

```

The results for various dictionaries and its combination does not have significant prediction using Naive Bayes model.

## B) SVM

- 1) Bing Dictionary - On using svm on the dataset we observe that the training data has a prediction success of approximately 81% and for test data the prediction success is approximately 77%, which implies that the combination of library and model is moderately fitted to the dataset.

```

user system elapsed
3.20 0.01 3.22
predicted
actual -1 1
-1 548 1056
1 91 4602
predicted
actual -1 1
-1 138 433
1 45 1484

```

- 2) AFINN Dictionary - On using svm on the dataset we observe that the training data has a prediction success of approximately 80% and for test data the prediction success is approximately 77%, which implies that the combination of library and model is moderately fitted to the dataset.

```

      user  system elapsed
      2.58    0.02   2.60
      predicted
actual   -1    1
      -1  504 1129
      1  120 4532
      predicted
actual   -1    1
      -1  131  417
      1    70 1477

```

- 3) NRC Dictionary - On using svm on the dataset we observe that the training data has a prediction success of approximately 85% and for test data the prediction success is approximately 77%, which implies that the combination of library and model is moderately fitted to the dataset.

```

      user  system elapsed
      773.59   24.36  814.72

Call:
best.tune(method = svm, train.x = as.factor(hiLo) ~ ., data = revDTM_sentiBing_trn
%>% select(-review_id),
  ranges = list(cost = c(0.1, 1, 10, 50), gamma = c(0.5, 1, 2, 5, 10)), kernel =
"radial")

Parameters:
  SVM-Type: C-classification
  SVM-Kernel: radial
  cost: 50

Number of Support Vectors: 3187

      predicted
actual   -1    1
      -1  930  699
      1  189 4430
      predicted
actual   -1    1
      -1  196  336
      1  141 1410

```

- 4) Combined Dictionary - On using svm on the dataset we observe that the training data has a prediction success of approximately 78% and for test data the prediction success is approximately 72%, which implies that the combination of library and model is moderately fitted to the dataset.

```

user  system elapsed
7.50    0.02   7.60
predicted
actual   -1    1
-1  636 1463
1   123 5102
predicted
actual   -1    1
-1  122  583
1   89   1648

```

Upon close analysis of the svm model with various dictionaries, the results are a moderate to predict the sentiments. These are better suited in comparison to naive bayes model.

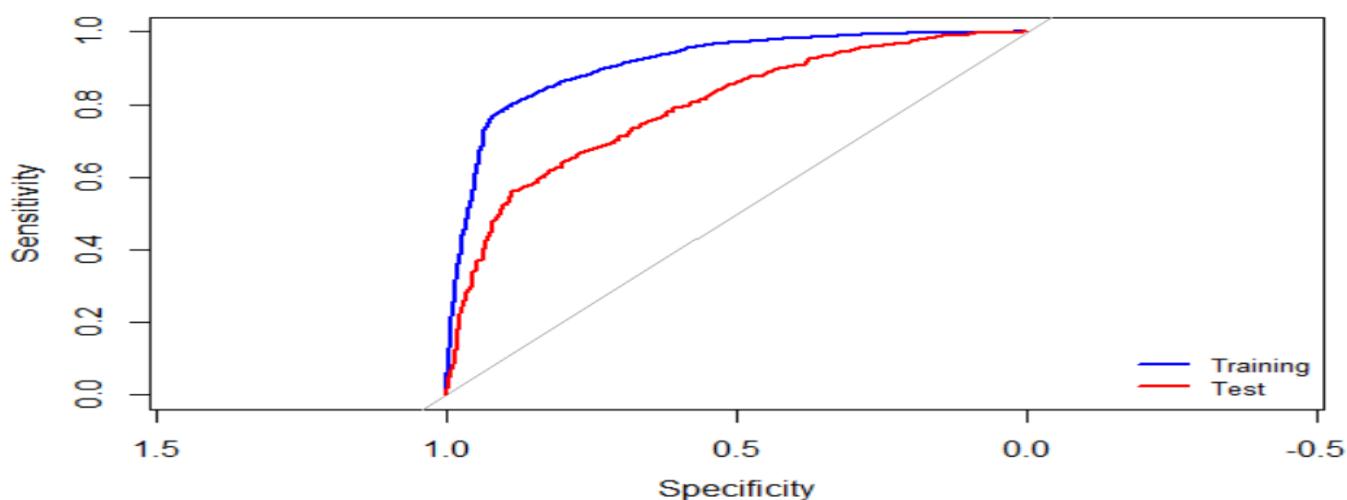
### C) Random Forest

- 1) Bing Dictionary - On using random forest on the dataset we observe that the training data has a prediction success of approximately 91% and for test data the prediction success is approximately 79%, which implies that the combination of library and model is well fitted to the dataset.

```

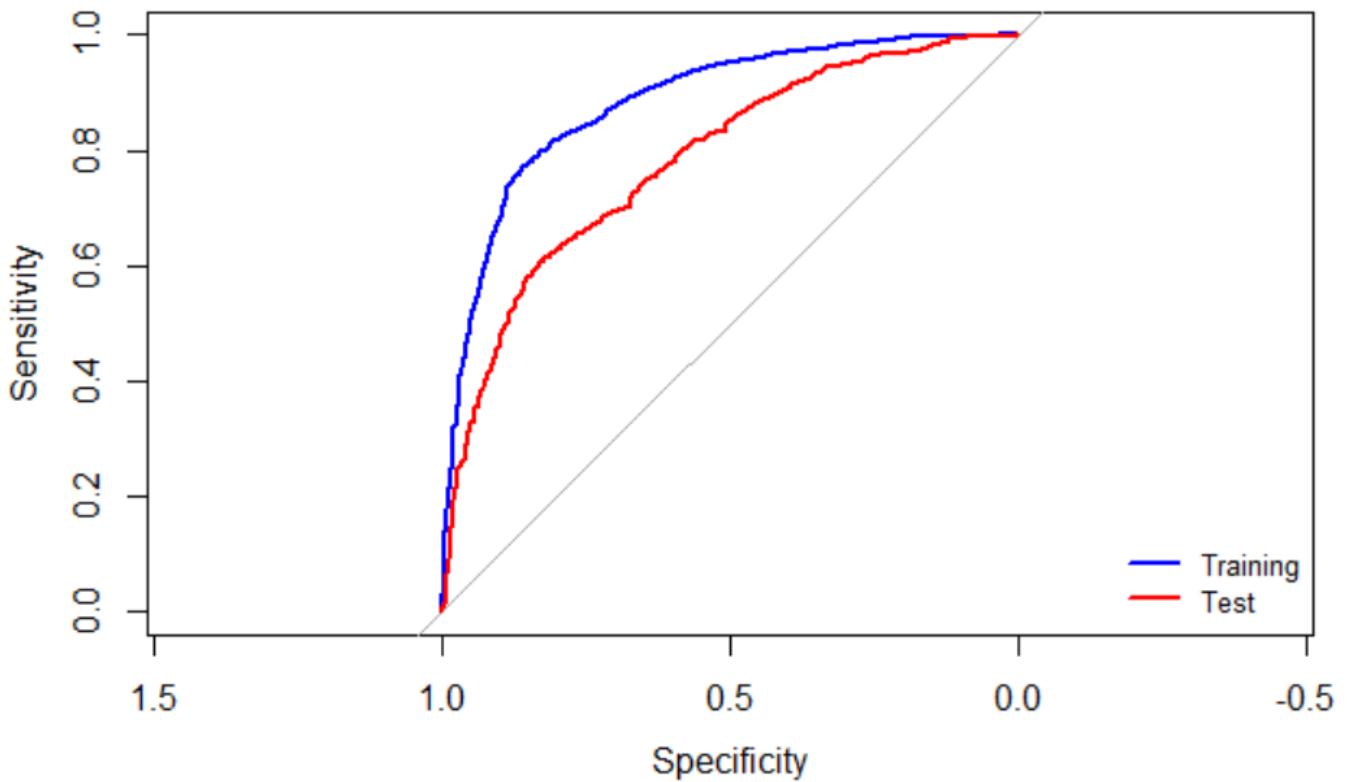
Computing permutation importance.. Progress: 35%. Estimated remaining time: 1 minute, 0 seconds.
Computing permutation importance.. Progress: 73%. Estimated remaining time: 23 seconds.
    preds
actual FALSE TRUE
-1 1203 401
1  533 4160
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.9121
    preds
actual FALSE TRUE
-1  289 282
1   222 1307
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7923

```



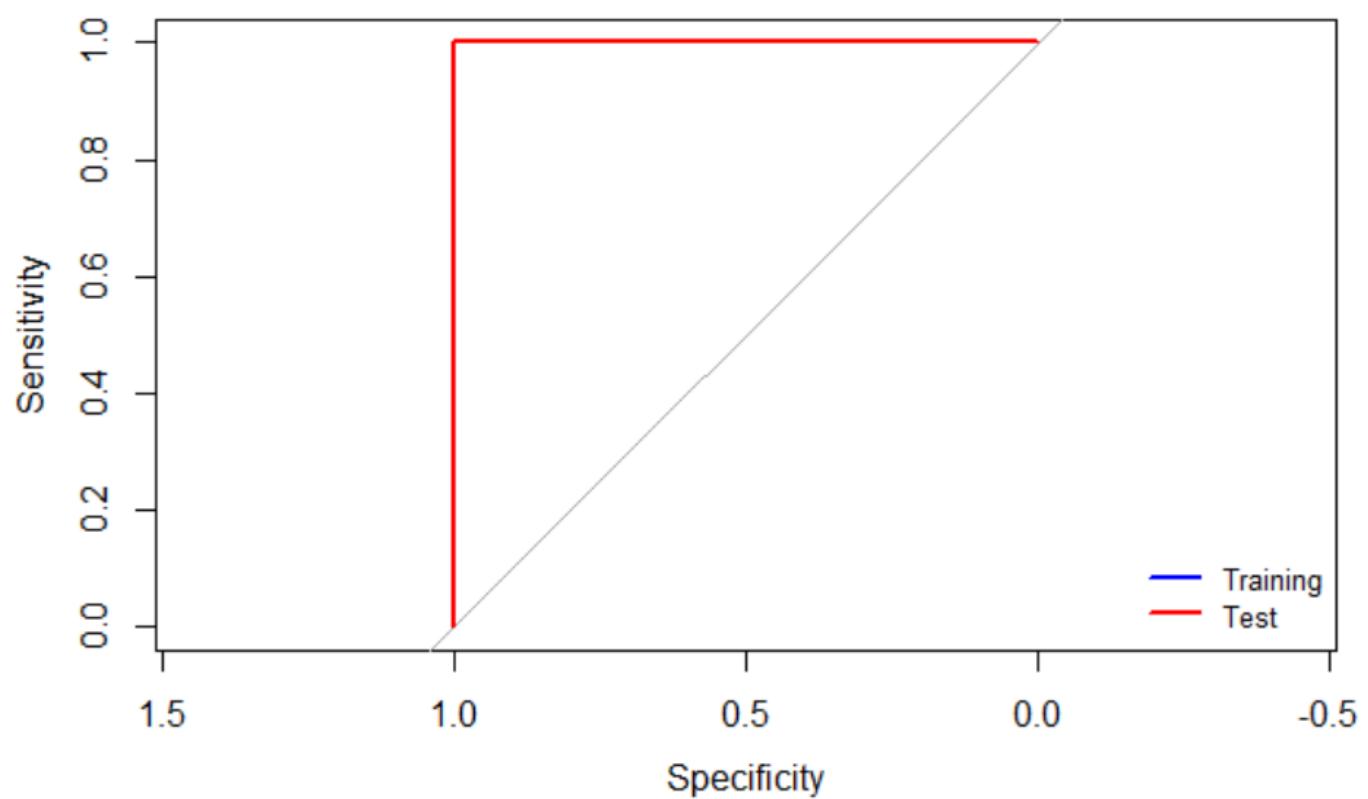
- 2) AFINN Dictionary - On using random forest on the dataset we observe that the training data has a prediction success of approximately 88% and for test data the prediction success is approximately 79%, which implies that the combination of library and model is moderately fitted to the dataset.

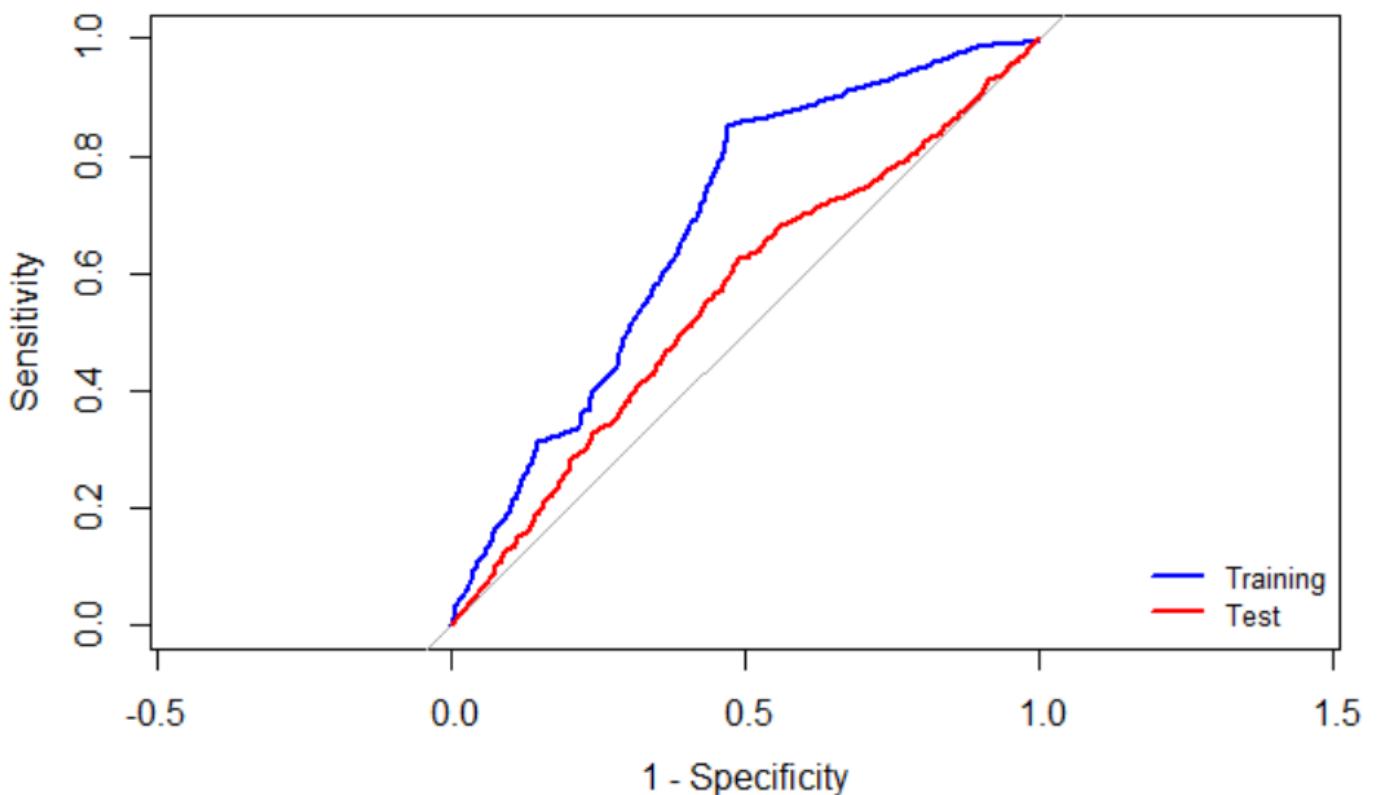
```
Computing permutation importance.. Progress: 100%. Estimated remaining time: 0 seconds.
preds
actual FALSE TRUE
-1 1168 465
1 617 4035
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.8847
preds
actual FALSE TRUE
-1 311 237
1 290 1257
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.7843
```



- 3) NRC Dictionary - On using random forest on the dataset we observe that the training data has a prediction success of approximately 1% and for test data the prediction success is approximately 1%, which implies that the combination of library and model is overly fitted to the dataset.

```
    preds
actual FALSE TRUE
  -1 2282   0
   1      0 5281
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 1
Adding missing grouping variables: `review_id`
    preds
actual FALSE TRUE
  -1 704   0
   1      0 1817
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 1
```



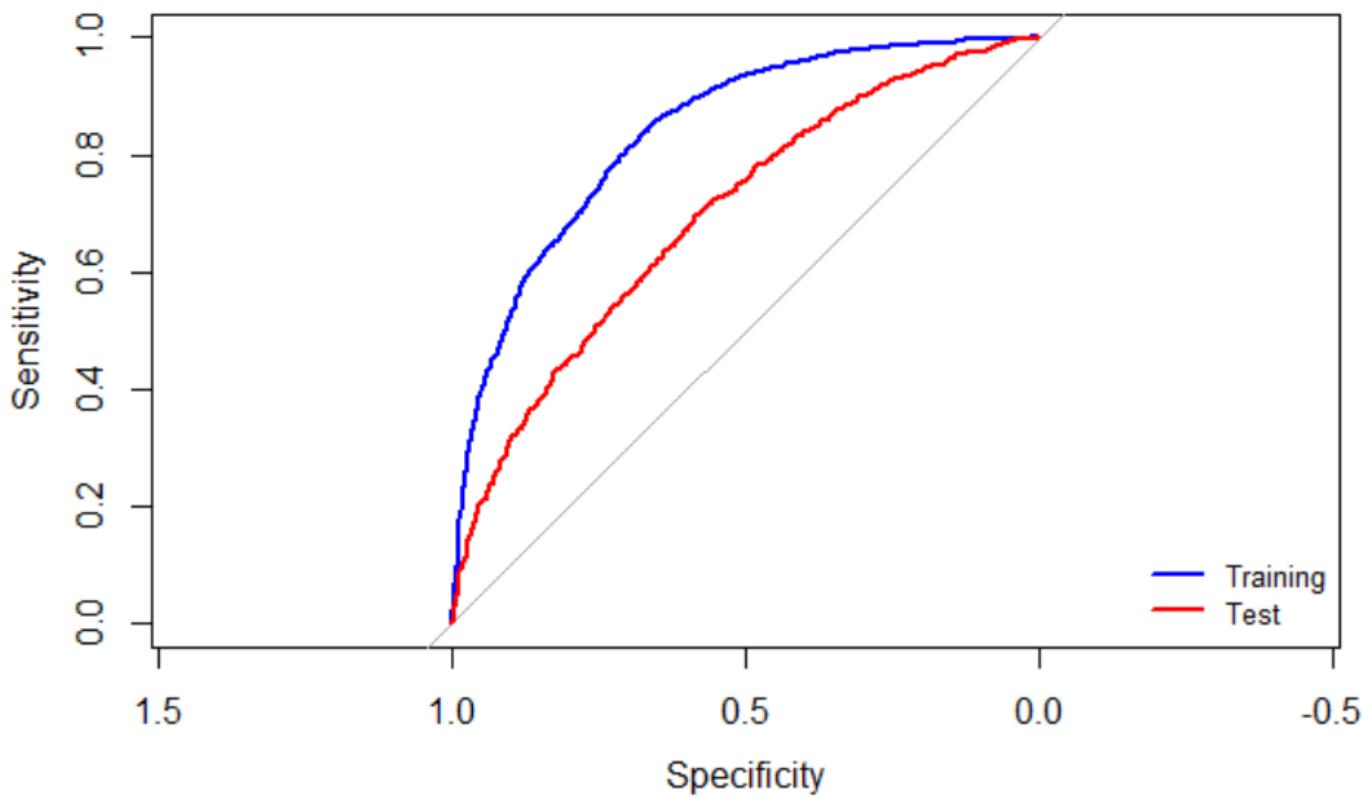


- 4) Combined Dictionary - On using random forest on the dataset we observe that the training data has a prediction success of approximately 84% and for test data the prediction success is approximately 69%, which implies that the combination of library and model is poorly fitted to the dataset.

```

Computing permutation importance.. Progress: 36%. Estimated remaining time: 55 seconds.
Computing permutation importance.. Progress: 74%. Estimated remaining time: 21 seconds.
    preds
actual FALSE TRUE
 -1 1371 728
  1 737 4488
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.8391
    preds
actual FALSE TRUE
 -1 302 403
  1 320 1417
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6971

```



The best predictions were so far achieved by random forest model over bing dictionary.

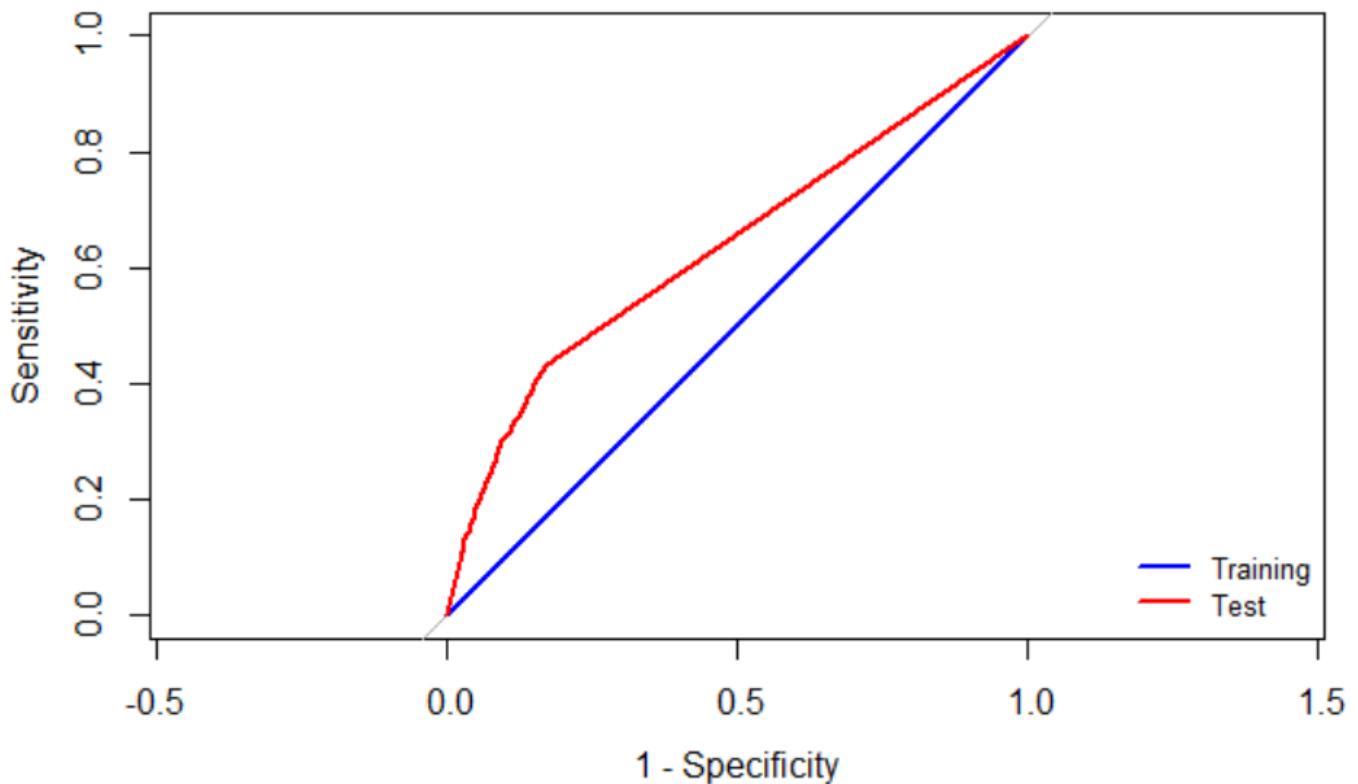
ii) ***Broader Terms:***

- A) Naive Bayes - On using naive bayes on the combined dictionary dataset we observe that the training data has a prediction success of approximately 50% and for test data the prediction success is approximately 64%, which implies that the combination of library and model is poorly fitted to the dataset.

```

predicted
actual TRUE
-1 1320
 1 3864
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.5004
Setting direction: controls < cases
predicted
actual FALSE TRUE
-1 1200  46
 1 3369  569
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.6488
Setting direction: controls < cases

```



- B) SVM - On using svm on the combined dictionary dataset we observe that the training data has a prediction success of approximately 100% and for test data the prediction success is approximately 80%, which implies that the combination of library and model is well fitted to the dataset.

```

user  system elapsed
34.02    2.21   36.80
      predicted
actual   -1    1
-1 1320    0
 1    0 3864
      predicted
actual   -1    1
-1  251   995
 1    22 3916

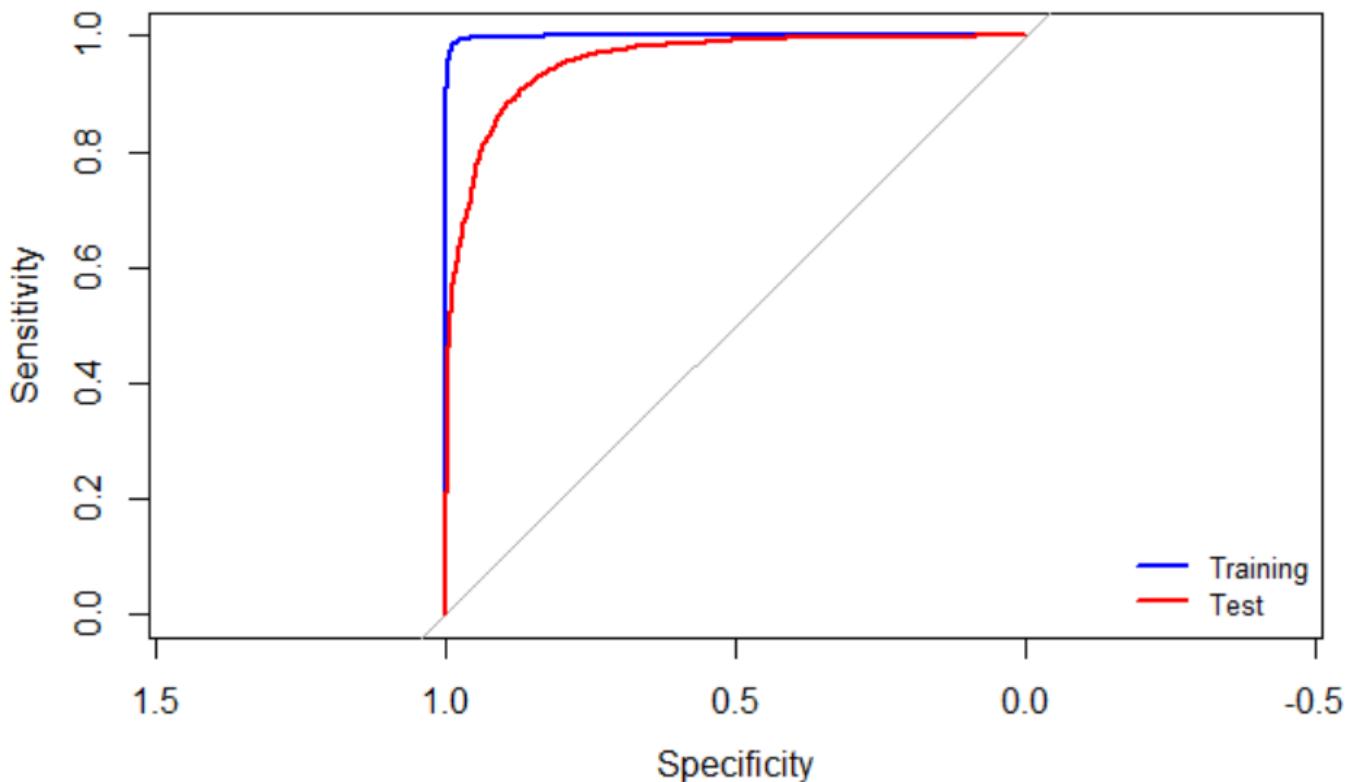
```

- C) Random Forest - On using random forest on the combined dictionary dataset we observe that the training data has a prediction success of approximately 99% and for test data the prediction success is approximately 95%, which implies that the combination of library and model is overly fitted to the dataset.

```

      preds
actual FALSE TRUE
-1 1306  14
 1 111 3753
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.9979
      preds
actual FALSE TRUE
-1 1063 183
 1 385 3553
Setting direction: controls < cases
Setting levels: control = 1, case = 2
Setting direction: controls < cases
Area under the curve: 0.949

```



The well-fitted model for ‘broader terms’ has been observed for svm model.

Q5. 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).

(i) 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.

(ii) 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.

Note: for question (e), you will consider the values in the ‘attribute’ column. This has values of multiple attributes, separated by a ‘|’. Further, some of the values, like Ambience, carry a list of True/False values (like, for example, Ambience: {'romantic': False, 'intimate': False, 'classy': False, 'hipster': False, ...}). Care must be taken to extract values for different attributes. You can consider a separate dataframe with review\_id, attribute, and then process this further to extract values for the different attributes.

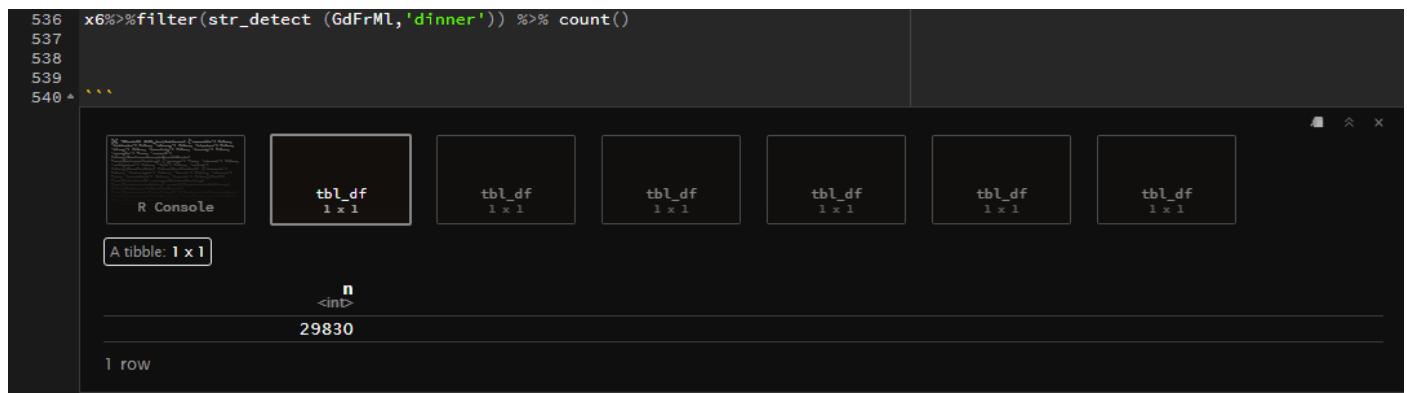
Ans. i) Summarization of few interesting attributes:

	GdFrMI	n
1	'dinner'	7919
2	character(0)	2355
3	'lunch'	5955
4	c(" 'dinner'", " 'brunch'")	531
5	c(" 'latenight'", " 'dinner'")	526
6	'latenight'	749
7	c(" 'lunch'", " 'dinner'", " 'brunch'")	48
8	{'dessert'}	869
9	c(" 'lunch'", " 'dinner'")	13259
10	c(" 'lunch'", " 'breakfast'")	405
11	'breakfast'	902
12	c(" {'dessert'", " 'lunch'", " 'dinner'", " 'breakfast'", " 'brunch'")	40
13	c(" 'latenight'", " 'lunch'", " 'dinner'")	1123
14	'brunch'	1009
15	c(" 'latenight'", " 'lunch'", " 'dinner'", " 'breakfast'")	94
16	c(" {'dessert'", " 'lunch'", " 'dinner'")	850

17	c(" 'lunch'", " 'dinner'", " 'breakfast'", " 'brunch'")	131
18	c(" 'latenight'", " 'breakfast'")	38
19	c(" {'dessert'", " 'latenight'", " 'breakfast'", " 'brunch'")	39
20	c(" 'lunch'", " 'breakfast'", " 'brunch'")	894
21	c(" {'dessert'", " 'lunch'", " 'breakfast'", " 'brunch'")	210
22	c(" {'dessert'", " 'lunch'", " 'dinner'", " 'brunch'")	78
23	c(" 'latenight'", " 'lunch'")	238
24	c(" 'latenight'", " 'lunch'", " 'breakfast'", " 'brunch'")	43
25	c(" {'dessert'", " 'lunch'", " 'breakfast'")	73
26	c(" 'dinner'", " 'breakfast'", " 'brunch'")	111
27	c(" {'dessert'", " 'lunch'", " 'dinner'", " 'breakfast'")	74
28	c(" {'dessert'", " 'dinner'")	77
29	c(" 'lunch'", " 'brunch'")	68
30	c(" 'breakfast'", " 'brunch'")	966
31	c(" {'dessert'", " 'brunch'")	48
32	c(" {'dessert'", " 'lunch'")	44

33	c(" 'lunch'", " 'dinner'", " 'breakfast'")	120
34	c(" '{dessert}'", " 'dinner'", " 'breakfast'", " 'brunch'")	78

```
536 x6 %>% filter(str_detect(GdFrMl,'dinner')) %>% count()  
537  
538  
539  
540 ▲ ``
```



Since, we observed that with “Broader terms” SVM worked well. Applying the same model to the dataset with attributes as parameters we can get a well fitted prediction.

