Aribnb_Review_Prediction

Mohit Kumar Maroti

10/14/2017

Airbnb Review Predictor Using Sentiment Analysis

Introduction:

Airbnb today has become the largest home-sharing network with over three million listings in 65,000+ cities spread over 191 countries. Certainly, the company has grown exponentially since 2008 and at the same time keeping the dual-sided market connected and happy. I've been using Airbnb since last three year and never had a bad experience. Finding a place to stay through Airbnb is very simple but sometimes the process can be tedious. "TO FIND THE RIGHT PLACE" a user has to filter out many options to find the right one.

Rating of a listing on Airbnb is widely used to determine the quality of the listing and sometimes to dig deep users tend to read the reviews before they make their decision. Though there are lots of wonderful hype about Airbnb, there are tons of pitfalls to using the service, including its lack of privacy, poor consistency and these get covered by ratings. Ratings can be deceptive sometimes as they are very difficult to describe the experience of a guest on a scale of 1 to 10. However, a review or a comment tell you a true story of guests experience. Reading all the reviews of listings can be time-consuming and if you have more listings to chose from the decision-making process can be tough and time-consuming.

Through this project, I'm trying to to find a way to predict an alternative review score and present the same to the user. The predictor score is called "Review Sentiment Score". The predictor score will be a true representation of user feeling and will present a true story behind a users experience.

The Airbnb Dataset:

I'm using Airbnb Asheville, NC listings dataset. This dataset has two files " Reviews.csv" and "Listings.csv". The listings datasets have different variables related to Airbnb listings and Reviews datasets have different variable related to reviews written against the listings. Raw data obtained for this project has a very well-defined structure. We have taken different approaches to clean the data. Apart from converting the data variables into the desired format for data analysis, we have taken steps to remove NA values from important variables.

Installing and loading the required packages for our analysis

```
wants <-
c('tidyverse','SentimentAnalysis','ggplot2','plotly','ggpubr','plyr','c
orrplot', 'randomForest','caTools','pROC','caret','e1071')
has <- wants %in% rownames(installed.packages())
if(any(!has)) install.packages(wants[!has])
sapply(wants, require, character.only = TRUE)
rm("wants","has")
```

Loading the datasets:

```
Reviews <- read.csv("reviews.csv", header= TRUE) #Loads the Reviews
file
Listings <- read.csv("listings.csv", header = TRUE) # Loads the
listings file
```

Structure of Datasets:

```
  str(Reviews)

## 'data.frame':    27721 obs. of  12 variables:
##  $ X            : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ listing_id   : int  665257 665257 665257 665257 665257 665257
665257 665257 665257 665257 ...
##  $ id           : int  2200401 2255411 2316293 2339727 2496120
2539714 2598913 2630086 2722753 2787556 ...
##  $ date         : Factor w/ 1463 levels "2010-10-28","2010-11-
30",..: 218 221 225 226 237 241 245 248 257 262 ...
##  $ reviewer_id  : int  3369543 799015 3395850 1491009 3416229
2653248 2414374 3400679 3694710 3915162 ...
##  $ reviewer_name: Factor w/ 5010 levels "(EMAIL HIDDEN)",..: 2202
506 2397 884 2385 1937 1492 2580 2148 505 ...
##  $ comments     : Factor w/ 27620 levels "","\n\nMy girfriends and i
really enjoyed staying in this lovely new home. Mark and Roses made
sure we were provided with the amen"| __truncated__,..: 5036 8826 17764
21070 12262 3225 16799 10993 8833 23332 ...
##  $ WordCount    : int  4 34 41 17 38 13 6 20 21 32 ...
##  $ SentimentGI  : num  0 0.206 0.244 0.471 0.289 ...
##  $ SentimentLM  : num  0 0.0588 0.0732 0.1176 0.0526 ...
##  $ SentimentQDAP: num  0.25 0.235 0.22 0.588 0.263 ...
##  $ SentimentHE  : num  0 0.0588 0.0488 0 0.0263 ...

  str(Listings)

## 'data.frame':    864 obs. of  95 variables:
##  $ id                              : int  665257 2746729 6919450
12286328 156926 3767793 5927700 6698737 7966916 2254521 ...
##  $ listing_url                     : Factor w/ 864 levels
"https://www.airbnb.com/rooms/10014090",..: 602 302 636 165 217 370 528
607 716 272 ...
##  $ scrape_id                       : num  2.02e+13 2.02e+13 2.02e+13
```

```
2.02e+13 2.02e+13 ...
##  $ last_searched                   : Factor w/ 2 levels
"4/18/16","4/19/16": 1 2 2 1 1 1 1 1 1 1 ...
##  $ last_scraped                    : Factor w/ 5 levels
"4/20/16","4/21/16",..: 3 3 3 3 3 3 3 3 3 2 ...
##  $ name                            : Factor w/ 859 levels "
Charming Sunny Cottage and Porch",..: 468 371 372 500 501 505 146 674
839 758 ...
##  $ summary                         : Factor w/ 813 levels "","    An
upstairs luxury apartment created in the upstairs of an original George
Vanderbilt cottage. Located in Historic Biltmore"| __truncated__,..:
806 454 100 560 1 221 754 393 766 239 ...
##  $ space                           : Factor w/ 744 levels "","
This cozy one bedroom apartment is in a perfect location for enjoying
everything that Asheville has to offer. We are minut"|
__truncated__,..: 721 387 1 1 140 213 86 495 79 150 ...
##  $ description                     : Factor w/ 862 levels "     This
cozy one bedroom apartment is in a perfect location for enjoying
everything that Asheville has to offer. We are minut"|
__truncated__,..: 855 482 105 596 233 232 798 419 810 254 ...
##  $ experiences_offered             : Factor w/ 1 level "none": 1 1 1
1 1 1 1 1 1 1 ...
##  $ neighborhood_overview           : Factor w/ 644 levels "","-My
neighborhood, typically called the East End, sits directly adjacent to
downtown. You can see downtown. You can walk there. "|
__truncated__,..: 1 6 1 1 1 415 57 497 594 397 ...
##  $ notes                           : Factor w/ 572 levels "","
Please park on the Elizabeth side. Also, the day of your arrival, a
general ETA is appreciated!      I have two small short-h"|
__truncated__,..: 1 366 1 1 1 1 454 551 505 130 ...
##  $ transit                         : Factor w/ 604 levels "","  3
miles to River Arts District,  4 miles to downtown 5 miles to the
Biltmore Estate 2 miles to French Broad River    Near 240,"|
__truncated__,..: 1 591 1 1 1 201 413 248 67 33 ...
##  $ thumbnail_url                   : Factor w/ 864 levels
"https://a0.muscache.com/im/pictures/100394974/98d287b4_original.jpg?ak
i_policy=small",..: 39 657 345 595 701 273 136 153 398 213 ...
##  $ medium_url                      : Factor w/ 864 levels
"https://a0.muscache.com/im/pictures/100394974/98d287b4_original.jpg?ak
i_policy=medium",..: 39 657 345 595 701 273 136 153 398 213 ...
##  $ picture_url                     : Factor w/ 864 levels
"https://a0.muscache.com/im/pictures/100394974/98d287b4_original.jpg?ak
i_policy=large",..: 39 658 346 596 702 274 136 153 397 214 ...
##  $ xl_picture_url                  : Factor w/ 864 levels
"https://a0.muscache.com/im/pictures/100394974/98d287b4_original.jpg?ak
i_policy=x_large",..: 39 657 345 595 701 273 136 153 398 213 ...
##  $ host_id                         : int  2600734 4295819 36223353
746673 746673 9385592 27672030 29622653 14686944 546222 ...
##  $ host_url                        : Factor w/ 643 levels
"https://www.airbnb.com/users/show/10032288",..: 231 402 348 593 593
```

```
632 258 277 73 499 ...
##  $ host_name                      : Factor w/ 464 levels "A Model
Community",..: 176 209 122 61 61 238 177 237 199 296 ...
##  $ host_since                     : Factor w/ 539 levels
"1/1/15","1/11/13",..: 343 119 360 378 378 49 160 222 279 286 ...
##  $ host_location                  : Factor w/ 42 levels "Arden,
North Carolina, United States",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ host_about                     : Factor w/ 492 levels "","  I am
a native of Charlotte N.C., have lived in Asheville for 20 years and
love it.\nSixteen years ago I joined with a partner"| __truncated__,..:
236 469 413 480 480 421 317 358 414 1 ...
##  $ host_response_time             : Factor w/ 5 levels "a few days
or more",..: 4 5 2 4 4 4 5 5 5 4 ...
##  $ host_response_rate             : Factor w/ 28 levels
"100%","33%","50%",..: 1 1 28 1 1 1 1 1 1 1 ...
##  $ host_acceptance_rate           : Factor w/ 50 levels
"0%","100%","20%",..: 50 34 50 20 20 24 2 46 2 2 ...
##  $ host_is_superhost              : Factor w/ 2 levels "f","t": 1 2
1 1 1 1 1 2 2 1 ...
##  $ host_thumbnail_url             : Factor w/ 643 levels
"https://a0.muscache.com/im/users/10077483/profile_pic/1389395025/origi
nal.jpg?aki_policy=profile_small",..: 58 599 90 283 283 138 558 565 505
619 ...
##  $ host_picture_url               : Factor w/ 643 levels
"https://a0.muscache.com/im/users/10077483/profile_pic/1389395025/origi
nal.jpg?aki_policy=profile_x_medium",..: 58 599 90 283 283 138 558 565
505 619 ...
##  $ host_neighbourhood             : Factor w/ 3 levels "","Glenvar
Heights",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ host_listings_count            : int  1 5 1 6 6 3 1 1 1 1 ...
##  $ host_total_listings_count      : int  1 5 1 6 6 3 1 1 1 1 ...
##  $ host_verifications             : Factor w/ 61 levels "['email',
'manual_online', 'facebook', 'reviews', 'manual_offline', 'kba']",..:
21 47 56 23 23 21 20 49 10 21 ...
##  $ host_has_profile_pic           : Factor w/ 2 levels "f","t": 2 2
2 2 2 2 2 2 2 2 ...
##  $ host_identity_verified         : Factor w/ 2 levels "f","t": 2 2
1 1 1 2 2 2 2 2 ...
##  $ street                         : Factor w/ 576 levels ",
Asheville, NC 28801, United States",..: 234 106 489 231 230 287 499 383
170 273 ...
##  $ neighbourhood                  : logi  NA NA NA NA NA NA ...
##  $ neighbourhood_cleansed         : int  28806 28806 28806 28806
28806 28806 28806 28806 28806 28806 ...
##  $ city                           : Factor w/ 7 levels
"Arden","Asheville",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ state                          : Factor w/ 1 level "NC": 1 1 1 1
1 1 1 1 1 1 ...
##  $ zipcode                        : int  28806 28806 28806 28806
28806 28806 28806 28806 28806 28806 ...
```

```
##  $ market                          : Factor w/ 1 level "North
Carolina Mountains": 1 1 1 1 1 1 1 1 1 ...
##  $ smart_location                  : Factor w/ 7 levels "Arden,
NC","Asheville, NC",..: 2 2 2 2 2 2 2 2 2 ...
##  $ country_code                    : Factor w/ 1 level "US": 1 1 1 1
1 1 1 1 1 ...
##  $ country                         : Factor w/ 1 level "United
States": 1 1 1 1 1 1 1 1 1 ...
##  $ latitude                        : num  35.6 35.6 35.6 35.6 35.6
...
##  $ longitude                       : num  -82.6 -82.6 -82.6 -82.6 -
82.6 ...
##  $ is_location_exact               : Factor w/ 2 levels "f","t": 2 1
1 2 2 2 2 2 2 ...
##  $ property_type                   : Factor w/ 17 levels
"Apartment","Bed & Breakfast",..: 10 13 10 8 10 10 10 10 10 2 ...
##  $ room_type                       : Factor w/ 3 levels "Entire
home/apt",..: 1 1 2 3 3 1 2 2 2 2 ...
##  $ accommodates                    : int  2 4 1 6 6 10 2 2 3 2 ...
##  $ bathrooms                       : num  1 1 1 2.5 2.5 2.5 1 1 1 1
...
##  $ bedrooms                        : int  1 1 1 1 1 4 1 1 1 1 ...
##  $ beds                            : int  1 2 1 6 6 5 1 1 2 1 ...
##  $ bed_type                        : Factor w/ 5 levels
"Airbed","Couch",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ amenities                       : Factor w/ 818 levels "{\"Air
Conditioning\",Kitchen,\"Free Parking on Premises\",\"Indoor
Fireplace\",Heating,\"Family/Kid Friendly\",\"Smoke Detecto"|
__truncated__,..: 65 496 579 203 250 398 178 77 58 271 ...
##  $ square_feet                     : int  NA NA NA NA NA NA NA NA NA
0 ...
##  $ price                           : Factor w/ 149 levels "$1,550.00
","$100.00 ",..: 4 19 103 65 65 4 122 129 129 122 ...
##  $ weekly_price                    : Factor w/ 182 levels
"","$1,000.00 ",..: 132 1 1 1 1 27 88 108 118 66 ...
##  $ monthly_price                   : Factor w/ 131 levels
"","$1,000.00 ",..: 30 1 1 1 1 99 1 39 1 117 ...
##  $ security_deposit                : Factor w/ 24 levels
"","$1,000.00 ",..: 4 1 1 1 1 14 4 1 1 1 ...
##  $ cleaning_fee                    : Factor w/ 54 levels "","$10.00
","$100.00 ",..: 15 32 25 1 1 16 1 37 15 2 ...
##  $ guests_included                 : int  2 2 1 1 1 4 2 1 2 1 ...
##  $ extra_people                    : Factor w/ 24 levels "$0.00
","$10.00 ",..: 2 1 1 10 10 5 6 1 8 6 ...
##  $ minimum_nights                  : int  7 2 1 1 1 2 1 1 2 1 ...
##  $ maximum_nights                  : int  120 1125 1125 7 7 180 1125
1125 7 1125 ...
##  $ calendar_updated                : Factor w/ 26 levels "1 week
ago","10 months ago",..: 18 4 11 25 11 25 25 23 10 8 ...
##  $ has_availability                : logi  NA NA NA NA NA NA ...
```

```
##  $ availability_30                : int  0 17 29 28 25 17 16 10 7 0
...
##  $ availability_60                : int  21 36 59 56 49 41 42 35 22
14 ...
##  $ availability_90                : int  51 57 89 86 78 67 68 62 39
27 ...
##  $ availability_365               : int  326 228 364 361 353 337
339 337 109 262 ...
##  $ calendar_last_scraped          : Factor w/ 5 levels
"4/20/16","4/21/16",..: 3 3 3 3 3 3 3 2 3 1 ...
##  $ number_of_reviews              : int  47 65 0 0 114 24 98 50 30
16 ...
##  $ first_review                   : Factor w/ 399 levels
"","1/1/15","1/11/15",..: 390 209 1 1 360 353 197 282 58 123 ...
##  $ last_review                    : Factor w/ 148 levels
"","1/1/15","1/1/16",..: 137 88 1 1 117 84 104 106 106 11 ...
##  $ review_scores_rating           : int  96 96 NA NA 93 78 93 98 99
94 ...
##  $ review_scores_accuracy         : int  10 9 NA NA 9 8 10 10 10 10
...
##  $ review_scores_cleanliness      : int  9 10 NA NA 9 8 10 10 10 9
...
##  $ review_scores_checkin          : int  10 10 NA NA 10 9 10 10 10
10 ...
##  $ review_scores_communication    : int  10 10 NA NA 9 9 10 10 10
10 ...
##  $ review_scores_location         : int  9 9 NA NA 9 8 9 10 10 10
...
##  $ review_scores_value            : int  10 9 NA NA 10 8 9 10 10 9
...
##  $ requires_license               : Factor w/ 1 level "f": 1 1 1 1
1 1 1 1 1 1 ...
##  $ license                        : logi  NA NA NA NA NA NA ...
##  $ jurisdiction_names             : Factor w/ 2 levels "NORTH
CAROLINA, NORTH CAROLINA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ instant_bookable               : Factor w/ 2 levels "f","t": 1 1
1 1 1 1 2 2 2 1 ...
##  $ cancellation_policy            : Factor w/ 4 levels
"flexible","moderate",..: 2 3 1 2 2 3 1 2 2 3 ...
##  $ require_guest_profile_picture  : Factor w/ 2 levels "f","t": 1 1
1 1 1 1 1 1 1 1 ...
##  $ require_guest_phone_verification: Factor w/ 2 levels "f","t": 1 1
1 1 1 1 1 1 1 1 ...
##  $ region_id                      : int  21 21 21 21 21 21 21 21 21
21 ...
##  $ region_name                    : Factor w/ 2 levels
"Buncombe","Henderson": 1 1 1 1 1 1 1 1 1 1 ...
##  $ region_parent_id               : int  37 37 37 37 37 37 37 37 37
37 ...
##  $ calculated_host_listings_count  : logi  NA NA NA NA NA NA ...
```

```
##  $ reviews_per_month               : num  1.06 2.69 NA NA 2.02 1.15
7.95 4.67 4.48 0.6 ...
```

Data wrangling

The two data files which I've used require some data initial wrangling in order to proceed with my analysis. In this step, I've done some basic data wrangling but as we move further I will be making some more changes in the variable format as an when required.

```
#step0: changing the column name so that I can join the table
colnames(Listings)[1] <- "listing_id"
#step1: Remove the listings which are not rated
drop_row <- which(is.na(Listings$review_scores_accuracy))
Listings <- Listings[-(drop_row),]

#steps 2 Remove the reviews which have no comments
which(is.na(Reviews$comments))

## integer(0)

#step 3 Remove if unique identifiers are missing
which(is.na(Reviews$listing_id))

## integer(0)

#step4: Cleaning the review file in the desired format
Reviews$comments <- as.character(Reviews$comments) #converting the
comments from factor to character type
```

In step0 I have renamed the first column of Listing to " listing_id" so that I can join it with the review file when required. In Step1 Step2 and Step3: I have tried to find which rows are empty and then remove them. In step2 and Step3: return value is 0 so there is no need to remove any observation. In step4: I have converted the Comments variable into character format which will be required for sentiment analysis of comments.

Exploring the Data:

```
summary(Listings)
summary(Reviews)
```

Counting Number of Reviews from the Reviews data file.

```
length(unique(Reviews$id))

## [1] 27721
```

Counting Number of Listings from the listing data file.

```
length(unique(Listings$listing_id))

## [1] 741
```

Listing distribution over different cities

```
length(unique(Listings$city))

## [1] 7
```

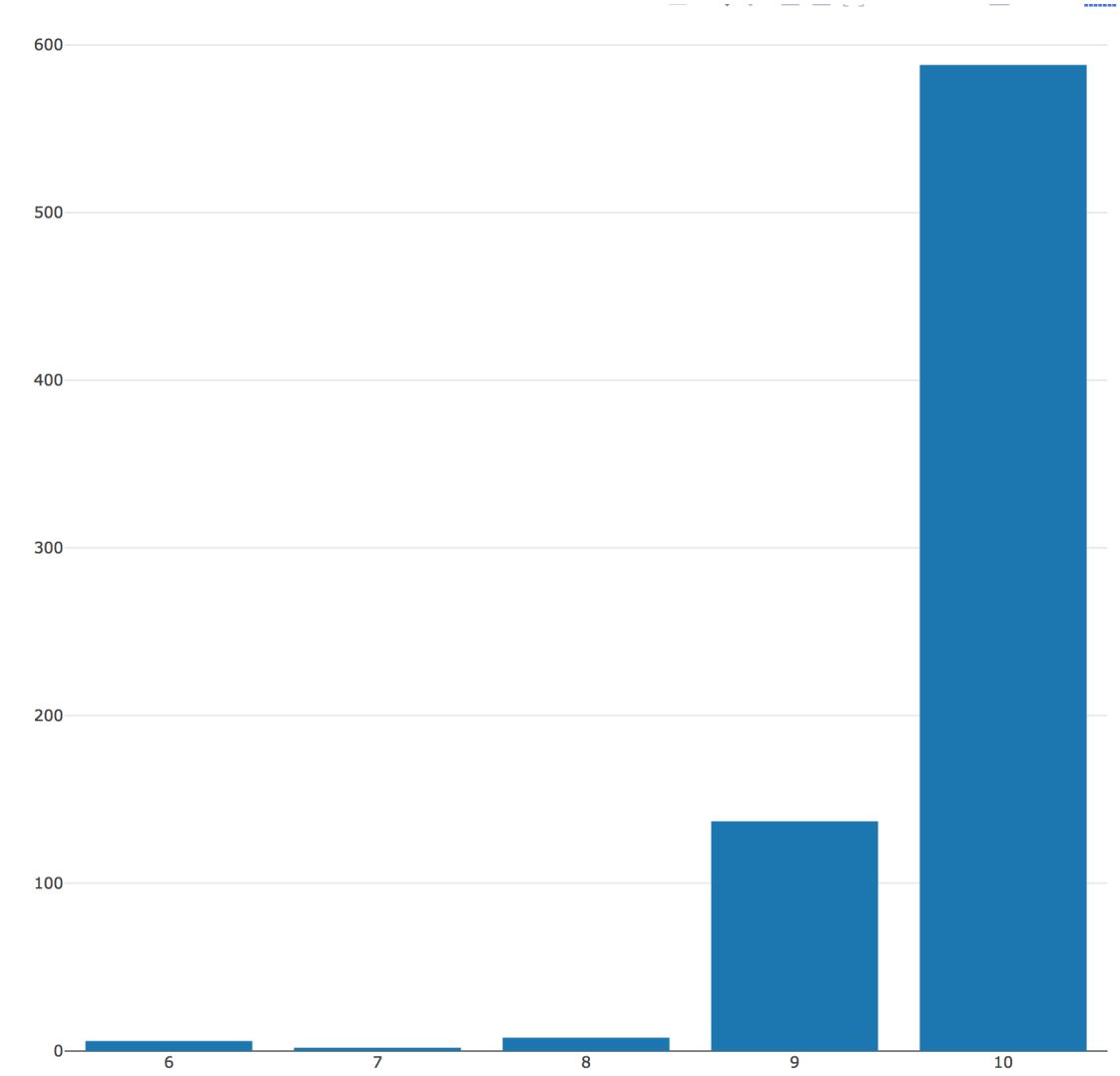
Here we can see that there are 27721 reviews written against 741 listings spread over 7 different cities in Asheville, NC.

Rating Score distribution

```
summary(Listings$review_scores_accuracy)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   6.000  10.000  10.000   9.753  10.000  10.000

ratings_freq <- as.data.frame(table(Listings$review_scores_accuracy))
plot1 <- plot_ly(x = ratings_freq$Var1,y = ratings_freq$Freq,type =
'bar')
```



We can see that minimum review score of a listing is 6 and max is 10 and mean

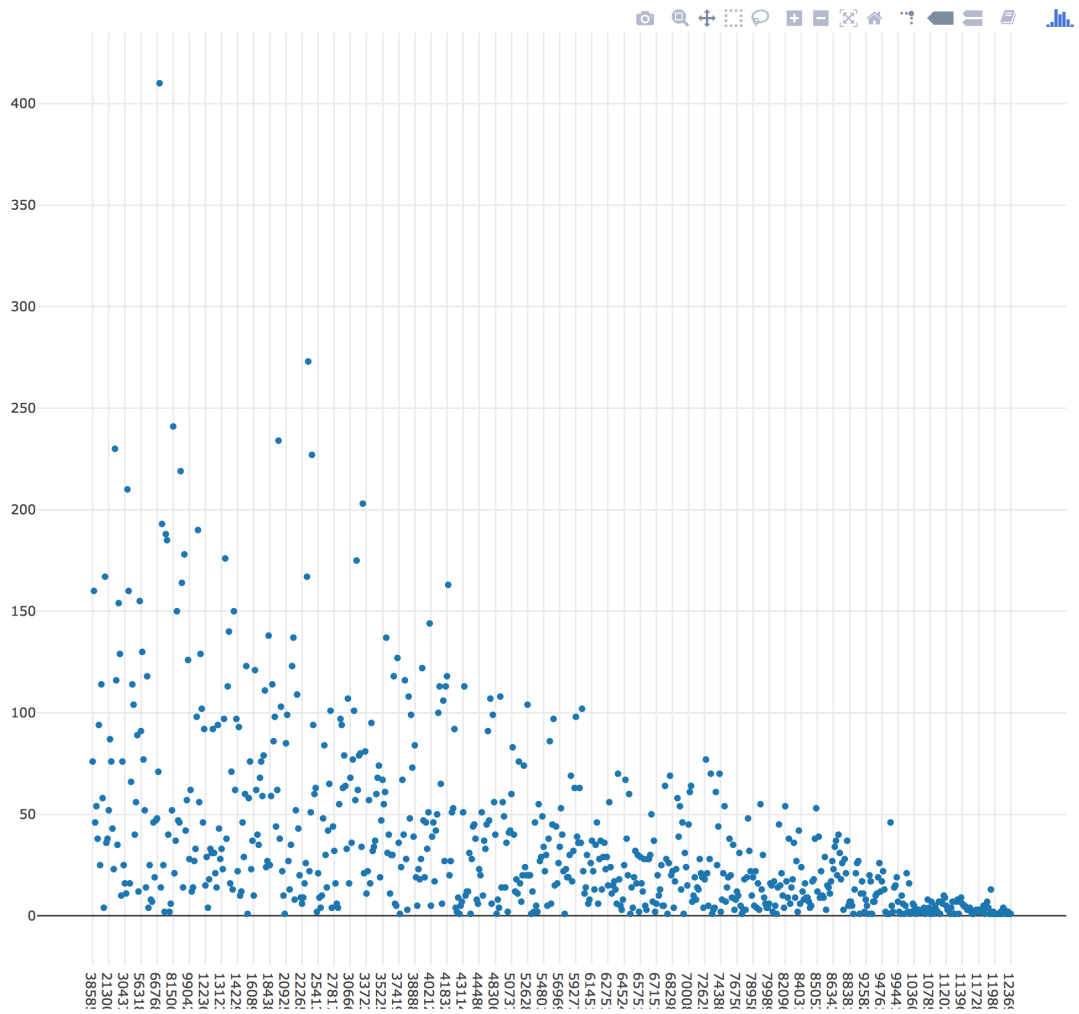
score is 9.75 score is which pretty high. This data tells us most of the listings are rated really high and ploty graph support that observation. Here you might think most of the ratings are positive then how is this helping the user? Well having many positive reviews adds more value to review sentiment score as it will help you pick best from the best.

Number of Reviews distribution

```
reviews_freq <- as.data.frame(table(Reviews$listing_id))
summary(reviews_freq$Freq)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    7.00   22.00   37.36   48.75  410.00

plot_ly(x = reviews_freq$Var1,y = reviews_freq$Freq,type = 'scatter')
```



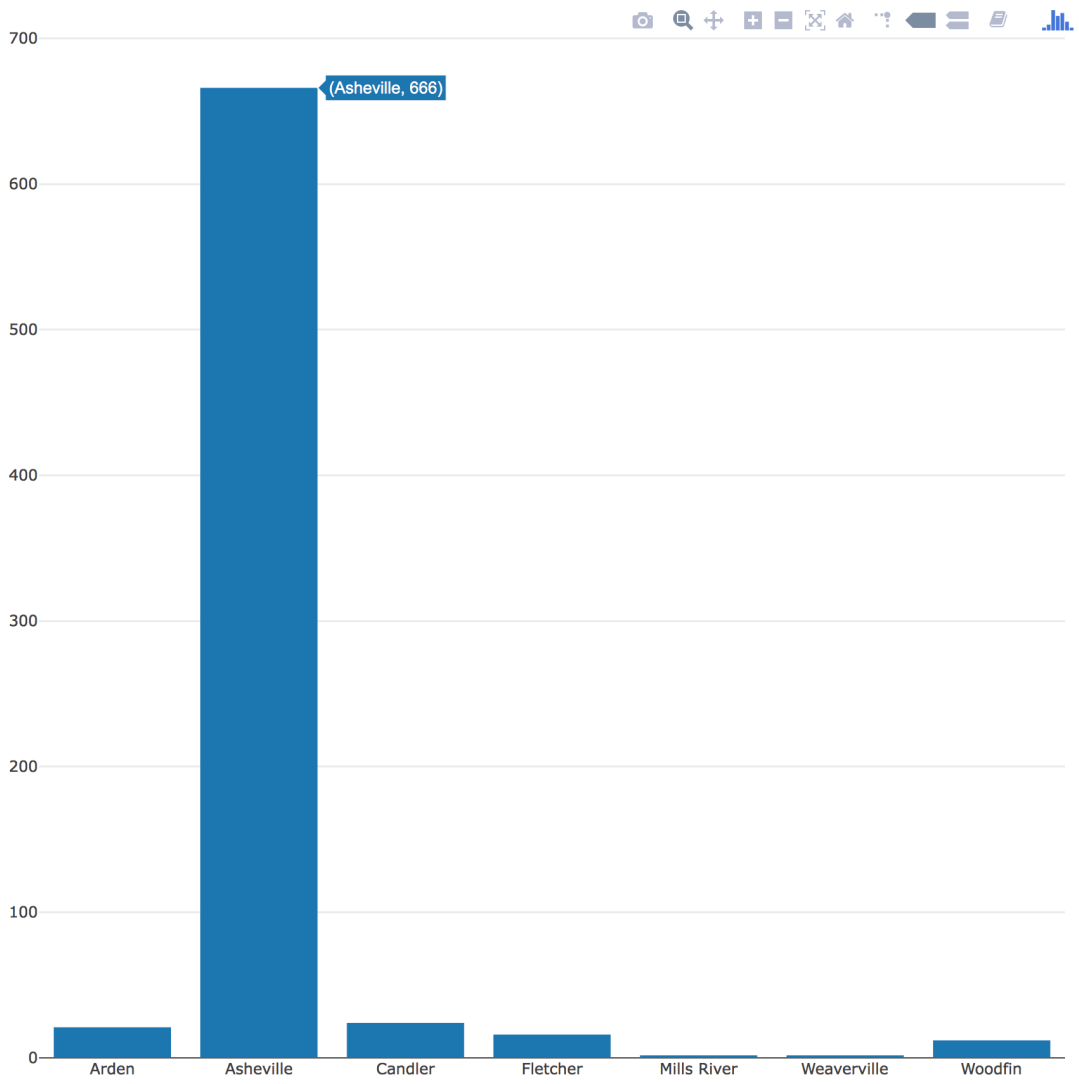
We can see here that all the listings are reviewed at least once and one or more review have been reviewed 410 times. The average review per listing is 37. The ploty graph will help you see the distribution more closely.

Listing Distribution over different cities

```
city_freq <- as.data.frame(table(Listings$city))
summary(city_freq$Freq)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     6.5    16.0   105.9    22.5   666.0

plot_ly(x = city_freq$Var1,y = city_freq$Freq,type = 'bar')
```



Here we are seeing the distribution of Airbnb listing over different cities.Clearly, we can see that Asheville city has the highest number of listing and Mills River and Weaverville have just 1 listing each.

Now that we have seen how the listings, Ratings, and reviews are distributed it's time to dig deep into reviews and apply sentiment analysis engine to predict the sentiment of each review.

Sentiment Analysis

For conducting sentiment analysis I have used SenetimentAnalysis Package. SentimentAnalysis performs a sentiment analysis of textual contents in R. This implementation utilizes various existing dictionaries, such as QDAP, Harvard IV or Loughran-McDonald.

The most important functions in SentimentAnalysis are:

1. Compute sentiment scores from contents stored in different formats with analyzeSentiment().

2. If desired, convert the continuous scores to either binary sentiment classes (negative or positive) or tertiary directions (negative, neutral or positive). This conversion can be done with convertToBinary() or convertToDirection() respectively.

Using this package we can represent the sentiment score to binary or tertiary directions but for this project, I have the raw sentiment score for the analysis.

Implementing the SentimentAnalysis Package

Setp1: Creating Variables in review to score all the sentiment scores

```
Reviews$WordCount<- NA
Reviews$SentimentGIs <- NA
Reviews$SentimentLMs<- NA
Reviews$SentimentQDAPs <- NA
Reviews$SentimentHEs <- NA
```

Step2: Running the sentiment engine

```
for(i in 1:nrow(Reviews) ){
  com <- Reviews$comments[i]
  Reviews_sentiment <- analyzeSentiment(com)
  Reviews$WordCount[i] <- Reviews_sentiment$WordCount
  Reviews$SentimentGI[i] <- Reviews_sentiment$SentimentGI
  Reviews$SentimentLM[i] <- Reviews_sentiment$SentimentLM
  Reviews$SentimentQDAP[i] <- Reviews_sentiment$SentimentQDAP
  Reviews$SentimentHE[i] <- Reviews_sentiment$SentimentHE
}
```

This is most time consuming process in process and it took my machine approximately 4 days to compute sentiment score for 27772 reviews. Now that we the we all reviews scored by the sentiment analysis enigne I have calculated the mean of all the three scores separtely for each unique listing_id and join then join it with the listing file.

```
Reviews_score<- aggregate(Reviews[, 7:11], list(Reviews$listing_id),
mean)
colnames(Reviews_score)[1] <- 'listing_id'
Listings_score <- join(Reviews_score, Listings, by = 'listing_id',
match = "all")
```

Let's look at how the review_score looks like:

```
head(Reviews_score)
```

```
  listing_id WordCount SentimentGI SentimentLM SentimentQDAP SentimentHE
1      38585  35.51316   0.2983558  0.06108250     0.3092891  0.03306302
2      67870  26.30625   0.2663403  0.05580324     0.2959882  0.02783974
3      80331  29.08696   0.2885931  0.08751471     0.3044215  0.04278813
4      80905  24.37037         NaN         NaN           NaN         NaN
5     136547  53.28947   0.2296739  0.06709461     0.2513653  0.02774733
6     155305  29.64894   0.1946362  0.07223019     0.2331715  0.02816647
```

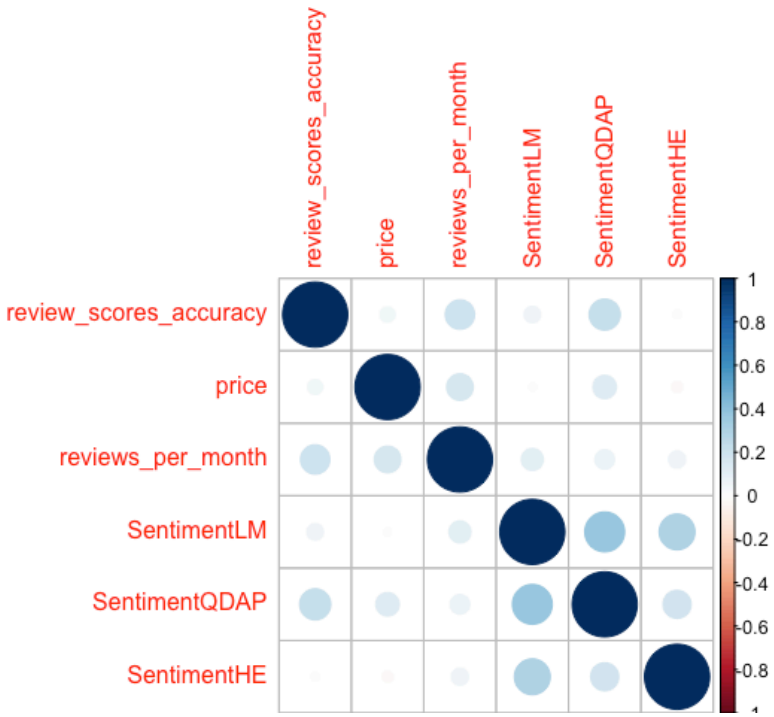
As you can this Review score table 6 variable: listing_id which is a unique identifier, SentimentGI, SentimentLM, SentimentQDAP and SentimentHE variables have the mean value of sentiment score generated by different dictionaries for each listing. For some of the listings the dictionaries the raw sentiment scores couldn't be generated as the dictionaries words are limited and also if the reviews are written in a language other than English the sentiment engine fails to generate a score.

Now let's join the review_scores

```
Listings_score <- join(Reviews_score, Listings, by = 'listing_id',
match = "all")
```

Let's see correlation between the variables which we will be using for our machine learning algorithm.

```
correlation_review <-
c("review_scores_accuracy","price","reviews_per_month",
                        "SentimentLM","SentimentQDAP","SentimentHE")
Correlation_review_df <- Listings_score[correlation_review]
Correlation_review_df$price <- as.numeric(Correlation_review_df$price)
Correlation_review_df$review_scores_accuracy <-
as.numeric(Correlation_review_df$review_scores_accuracy)
Correlation_review_df <- na.omit(Correlation_review_df)
Correlation_matrix <- cor(Correlation_review_df)
corrplot(Correlation_matrix)
```



Here we see some correlation between the target variable review_scores_accuracy and review scores generated by sentiment engine. Among all the sentiment score SentimentQDAP shows relatively high correlation with the target variable.

Feature Engineering:

The Listing_score dataframe have all our variables but all the variables. But for the analysis, I only need some feature variable. **Our target variable is the "review_scores_accuracy"** and other **important vaiables which are used for prediction are "SentimentLM","SentimentQDAP","SentimentHE","SentimentGI"** . In order **to improve the accuracy of the model I might use some more variables like:"listing_id","review_scores_cleanliness","review_scores_accuracy","review_scores_cleanliness","review_scores_checkin","review_scores_communication", "review_scores_location","review_scores_value", "instant_bookable","cancellation_policy","host_is_superhost","host_total_listings_count","neighbourhood_cleansed".**

Data Wrangling before applying the model

```
Listings_score$host_response_rate <-
gsub('%','',Listings_score$host_response_rate)
Listings_score$host_response_rate<-
as.numeric(Listings_score$host_response_rate)
Listings_score$host_response_rate<-
(Listings_score$host_response_rate)/100
```

```
Listings_score$review_scores_accuracy<-
as.factor(Listings_score$review_scores_accuracy)
```

Here I have converted the some of the variable to a format suitable to my model.Now I'll be creating a clean dataframe consisting of feature variables and with no NA values.

```
Listings_score_clean <- Listings_score[,c("listing_id"
,"WordCount","SentimentGI" ,"SentimentLM",

"SentimentQDAP","SentimentHE","review_scores_cleanliness",

"review_scores_accuracy","review_scores_cleanliness",

"review_scores_checkin","review_scores_communication",

"review_scores_location","review_scores_value",

"instant_bookable","cancellation_policy","host_is_superhost","host_tota
l_listings_count",
                                              "neighbourhood_cleansed")]
Listings_score_clean<- na.omit(Listings_score_clean)
```

Machine Learning

The model which I'm going to build is going to solve a classification problem i.e we are going to predict which class does the sentiment score belong to? given a scale of 1-10. I'm going to implement Random Tree and decision tree model to solve this classification problem.

Randon Forest

Random Forest is one of the most popular and most powerful machine learning algorithms. It is a type of ensemble machine learning algorithm called Bootstrap Aggregation or bagging.

The bootstrap is a powerful statistical method for estimating a quantity from a data sample. Such as a mean. You take lots of samples of your data, calculate the mean, then average all of your mean values to give you a better estimation of the true mean value.

In bagging, the same approach is used, but instead for estimating entire statistical models, most commonly decision trees.

Multiple samples of your training data are taken then models are constructed for each data sample. When you need to make a prediction for new data, each model makes a prediction and the predictions are averaged to give a better estimate of the true output value.

Random forest is a tweak on this approach where decision trees are created so that rather than selecting optimal split points, suboptimal splits are made by introducing randomness.

The models created for each sample of the data are therefore more different than they otherwise would be, but still accurate in their unique and different ways. Combining their predictions results in a better estimate of the true underlying output value.

Random Forest Implementation

Step1: Load all the libraries

```
library(randomForest)
library(caTools)
library(pROC)
library(caret)
```

Let's set the seed so that we get the same result

```
set.seed(123)
```

Step2: Splitting the data into test and training data

```
# our target variable is review_score_accuracy
rating <- sample.split(Y = Listings_score_clean$review_scores_accuracy,
SplitRatio = 0.65)
train_rating <- Listings_score_clean[rating,]
test_rating <- Listings_score_clean[!rating,]
```

Here I've added 65% of data into training dataset and remaining into test data.
Step3: Apply the randomforest

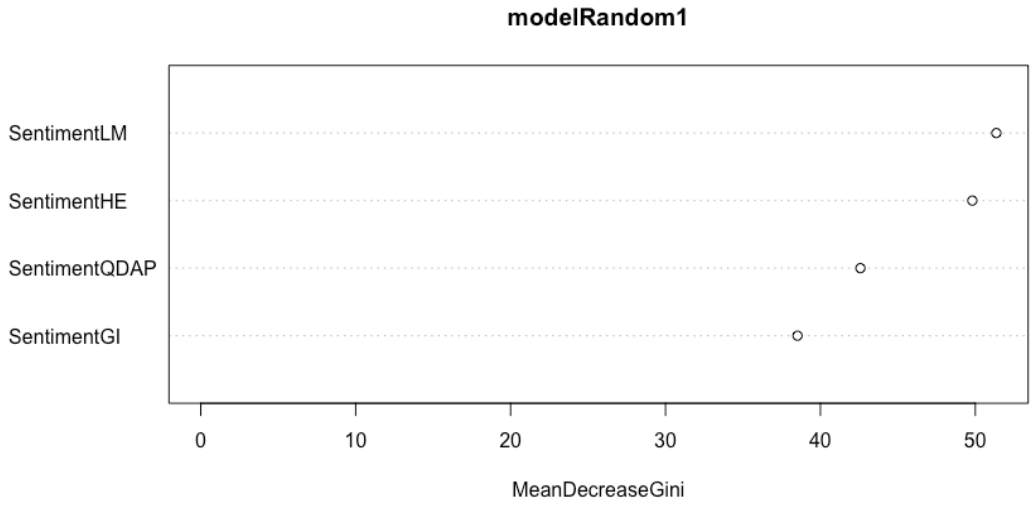
```
# high strength of tree = low error of individual tree classifier
#random forest parameters
#mtry = number of variables selected at each split (need to optimize)
# regression = floor (number of variables/3)
# categorical = floor(sqrt(no. of independent variables ))
# nodesize =  minimum size of terminal nodes
# Model1 Has only sentiment scores
modelRandom1 <- randomForest(review_scores_accuracy ~SentimentGI +
SentimentLM +
                               SentimentQDAP + SentimentHE+ , data =
train_rating, mtry =4, ntree= 2000)
modelRandom1
```

```
Call:
 randomForest(formula = review_scores_accuracy ~ SentimentGI +      SentimentLM + SentimentQDAP + SentimentHE, data = train_rating,      mtry = 4, ntree = 2000)
               Type of random forest: classification
                     Number of trees: 2000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 21.35%
Confusion matrix:
   6 7 8  9  10 class.error
6  0 0 0  0   4  1.00000000
7  0 0 0  0   2  1.00000000
8  0 0 0  1   5  1.00000000
9  0 0 0 11  89  0.89000000
10 2 0 0 14 420  0.03669725
```

We can see here OOB estimate of error rate: 21.35%. i.e out of bag rate is not that high. This implies the model is a good model and misclassification rate is not high. Let's see which variable is more important in the model

```
importance(modelRandom1)
varImpPlot(modelRandom1)
# higher MeanDecreaseGini more relevant the variable is.
```



Let's measure the accuracy of the model

```
PredictionWithClass <- predict(modelRandom1, test_rating, type =
'class')
t <- table(predictions= PredictionWithClass, actual =
test_rating$review_scores_accuracy)
sum(diag(t)/sum(t))
[1] 0.8365759
# the model has 83% accuracy
```

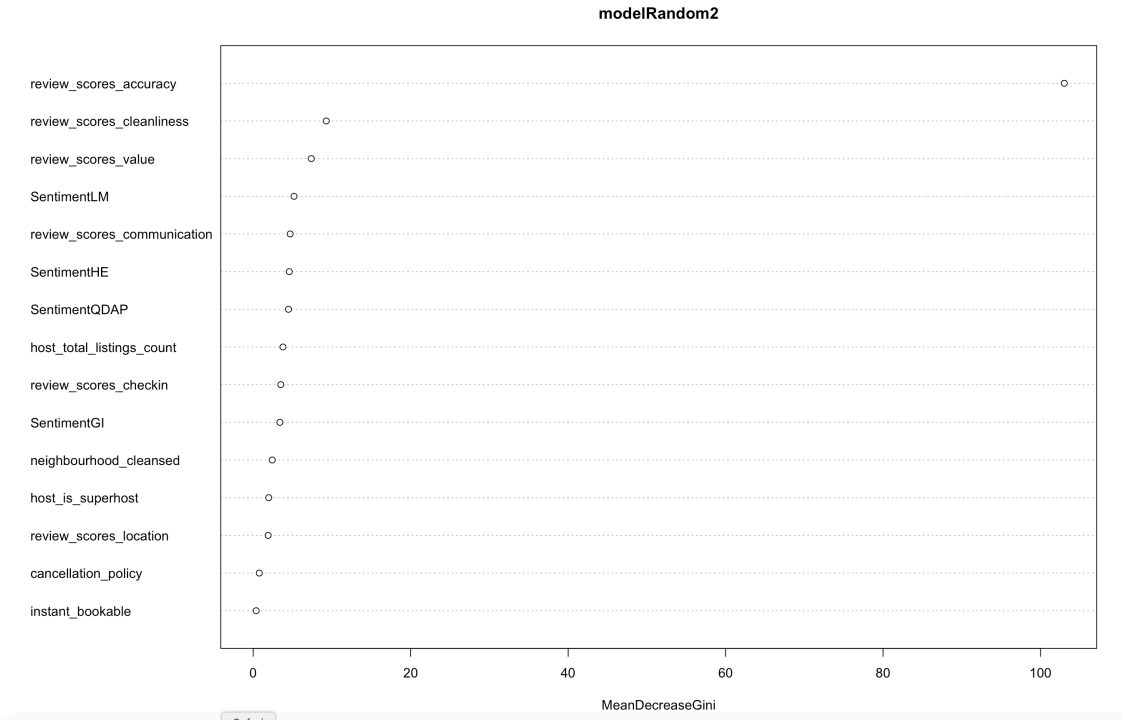
Step4: Apply the randomforest model2

```
#model2
modelRandom2 <- randomForest(review_scores_accuracy ~SentimentGI +
SentimentLM +
                               SentimentQDAP + SentimentHE +
instant_bookable + cancellation_policy +
                              host_is_superhost +
host_total_listings_count +
                              neighbourhood_cleansed +
review_scores_cleanliness +
                             review_scores_accuracy +
review_scores_cleanliness +
                             review_scores_checkin +
review_scores_communication +
                             review_scores_location, data =
```

```
train_rating, mtry =4, ntree= 2000)
modelRandom2
```

```
        OOB estimate of  error rate: 1.27%
Confusion matrix:
   6 7 8  9  10 class.error
6  1 0 1  1   1        0.75
7  0 0 0  0   1        1.00
8  0 0 3  1   1        0.40
9  0 0 0 86   0        0.00
10 0 0 0  0 378        0.00
```

```
# OOB estimate of  error rate: 1.27% is very low. i.e out of bag rate
is high
importance(modelRandom2)
varImpPlot(modelRandom2)
# higher MeanDecreaseGini more releavant it is.
```




```
PredictionWithClass1 <- predict(modelRandom2, test_rating, type =
'class')
t <- table(predictions= PredictionWithClass1, actual =
test_rating$review_scores_accuracy)
sum(diag(t)/sum(t))
# 98.7% accuracy
```

Though this model has high accuracy rate the predicator variable which are used to predict the target variable are related to each other. The model has endogenous variable like:review_scores_cleanliness,review_scores_accuracy,review_scores_cleanliness,review_scores_checking,review_scores_communication and review_scores_location which leads to endogeneity problem.

Decision Tree Implementation

Step1: Load all the libraries

```
install.packages('rpart')
install.packages('rpart.plot')
library(rpart)
library(rpart.plot)

#Let's set the seed so that we get the same result

set.seed(123)
```

Step2: Splitting the data into test and training data

```
# our target variable is review_score_accuracy
rating <- sample.split(Y = Listings_score_clean$review_scores_accuracy,
SplitRatio = 0.65)
train_rating <- Listings_score_clean[rating,]
test_rating <- Listings_score_clean[!rating,]
```

Here I've added 65% of data into training dataset and remaining into test data.

Step3: Fitting the model

```
modelDecisionTreeModel1<- rpart(review_scores_accuracy ~ SentimentGI +
SentimentQDAP + SentimentHE + SentimentLM
                                , data = train_rating, method =
'class')
plot(modelDecisionTreeModel1)
rpart.plot(modelDecisionTreeModel1)
summary(modelDecisionTreeModel1)

## Predictions
PredictionWithClass1 <- predict(modelDecisionTreeModel1, test_rating,
type = 'class')
table <- table(predictions=PredictionWithClass1, actual =
test_rating$review_scores_accuracy)
## Accuray Metric
sum(diag(table))/sum(table)
## 77 % accuracy
```

Conclusion

From above results we can conclude that random forest are more effective to predict the target variable. In randomforest models I would like to propose model1 as the model2 includes endogenous variable.

I've also implemented this model on New York Airbnb dataset and the results were very similar. The dataset had 60k listings and 666k reviews in it. Since the dataset was very large it would have taken a lot of time to compute the sentimental score. In order to procced with this dataset I had taken a subset containting 10% of the review data file and executed the same model. The result were very similar and random forest showed 79% accuracy whereas decision tree showed 58% accurracy. For this model to work on larger data we need to apply Bigdata techniques or microservice architecture for faster results.

Sentiment score can be used as an alternative score next to actual reviews and if the user sees a difference then they can refer to the review. This will save a lot of users time and make decision making easier.

The model fails to provide a score if there are no reviews written against a listing and sadly the model provides an inaccurate score for review with sarcasm in it.

Next

I would like to have a UI built on this model so that user can see both the actual score and sentiment score next to each other and if the user observes a difference in rating then they can read the reviews to find more details. I think the model can constantly improve over time with user feedback about the sentiment score. Also, wordcloud of reviews for each listing can be helpful visual for a user to visualize the negative and positive words in reviews.

THANK YOU