Report on Airbnb

Dhruv Jain & Venkata Dhanush Kikkisetti

2023-12-05

Contents

---

1 1. Loading the Data.

Let us see the Number of NA values from each column on the dataset. There are a lot of columns that have NA values in them. We will have replace them either with their median or remove the rows with NA values. Before we do that, we have to split the data to train and test set. Then We have to do transformation both on the trains set and test set.

```
## Rows: 38,792
## Columns: 75
## $ id                                  <dbl> 9.630344e+06, 3.533741e+0~
## $ listing_url                         <chr> "https://www.airbnb.com/r~
## $ scrape_id                           <dbl> 2.02e+13, 2.02e+13, 2.02e~
## $ last_scraped                        <chr> "02/10/23", "02/10/23", "~
## $ source                              <chr> "city scrape", "previous ~
## $ name                                <chr> "Rental unit in Brooklyn ~
## $ description                         <chr> "Enjoy your own private b~
## $ neighborhood_overview               <chr> "Easy access to subway. T~
## $ picture_url                         <chr> "https://a0.muscache.com/~
## $ host_id                             <dbl> 47783628, 17791294, 50213~
## $ host_url                            <chr> "https://www.airbnb.com/u~
## $ host_name                           <chr> "Tiffannie", "Taylor", "M~
## $ host_since                          <chr> "30/10/15", "07/07/14", "~
## $ host_location                       <chr> "New York, NY", "New York~
## $ host_about                          <chr> NA, NA, NA, "Lawyer / mus~
## $ host_response_time                  <chr> "N/A", "N/A", "N/A", "N/A~
## $ host_response_rate                  <chr> "N/A", "N/A", "N/A", "N/A~
## $ host_acceptance_rate                <chr> "N/A", "N/A", "N/A", "N/A~
## $ host_is_superhost                   <lgl> FALSE, FALSE, FALSE, FALS~
## $ host_thumbnail_url                  <chr> "https://a0.muscache.com/~
## $ host_picture_url                    <chr> "https://a0.muscache.com/~
## $ host_neighbourhood                  <chr> NA, "Hell's Kitchen", "Su~
## $ host_listings_count                 <dbl> 1, 1, 1, 1, 1, 2, 2, 1, 1~
## $ host_total_listings_count           <dbl> 2, 1, 1, 2, 1, 9, 9, 2, 2~
## $ host_verifications                  <chr> "['email', 'phone']", "['~
## $ host_has_profile_pic                <lgl> TRUE, TRUE, TRUE, TRUE, T~
## $ host_identity_verified              <lgl> FALSE, TRUE, FALSE, TRUE,~
## $ neighbourhood                       <chr> "Brooklyn , New York, Uni~
## $ neighbourhood_cleansed              <chr> "Bushwick", "Hell's Kitch~
## $ neighbourhood_group_cleansed        <chr> "Brooklyn", "Manhattan", ~
## $ latitude                            <dbl> 40.68457, 40.76878, 40.74~
## $ longitude                           <dbl> -73.91181, -73.98719, -73~
## $ property_type                       <chr> "Private room in rental u~
## $ room_type                           <chr> "Private room", "Private ~
## $ accommodates                        <dbl> 1, 2, 1, 2, 4, 1, 1, 2, 2~
## $ bathrooms                           <lgl> NA, NA, NA, NA, NA, NA, N~
## $ bathrooms_text                      <chr> "1 shared bath", "1 bath"~
## $ bedrooms                            <dbl> NA, 1, NA, 1, 1, NA, NA, ~
## $ beds                                <dbl> 1, 1, 1, 1, 2, 1, 1, 1, 1~
## $ amenities                           <chr> "[\"Body soap\", \"Dryer\~
## $ price                               <chr> "$65.00", "$110.00", "$99~
## $ minimum_nights                      <dbl> 30, 30, 30, 45, 30, 1, 1,~
## $ maximum_nights                      <dbl> 30, 1125, 1125, 1125, 180~
## $ minimum_minimum_nights              <dbl> 30, 30, 30, 45, 30, 1, 1,~
## $ maximum_minimum_nights              <dbl> 30, 30, 30, 45, 30, 1, 1,~
```

```
## $ minimum_maximum_nights                        <dbl> 30, 1125, 1125, 1125, 112~
## $ maximum_maximum_nights                        <dbl> 30, 1125, 1125, 1125, 112~
## $ minimum_nights_avg_ntm                        <dbl> 30, 30, 30, 45, 30, 1, 1,~
## $ maximum_nights_avg_ntm                        <dbl> 30, 1125, 1125, 1125, 112~
## $ calendar_updated                              <lgl> NA, NA, NA, NA, NA, NA, N~
## $ has_availability                              <lgl> TRUE, FALSE, TRUE, TRUE, ~
## $ availability_30                               <dbl> 29, 0, 0, 0, 3, 12, 5, 0,~
## $ availability_60                               <dbl> 59, 0, 0, 0, 20, 38, 31, ~
## $ availability_90                               <dbl> 89, 0, 0, 0, 35, 68, 58, ~
## $ availability_365                              <dbl> 364, 0, 0, 0, 96, 343, 33~
## $ calendar_last_scraped                         <chr> "02/10/23", "02/10/23", "~
## $ number_of_reviews                             <dbl> 5, 0, 0, 2, 88, 61, 41, 3~
## $ number_of_reviews_ltm                         <dbl> 0, 0, 0, 0, 14, 31, 9, 0,~
## $ number_of_reviews_l30d                        <dbl> 0, 0, 0, 0, 0, 1, 3, 0, 0~
## $ first_review                                  <chr> "14/08/16", NA, NA, "07/0~
## $ last_review                                   <chr> "30/04/18", NA, NA, "20/0~
## $ review_scores_rating                          <dbl> 4.80, NA, NA, 5.00, 4.98,~
## $ review_scores_accuracy                        <dbl> 4.60, NA, NA, 5.00, 5.00,~
## $ review_scores_cleanliness                     <dbl> 5.00, NA, NA, 5.00, 4.99,~
## $ review_scores_checkin                         <dbl> 5.00, NA, NA, 5.00, 4.95,~
## $ review_scores_communication                   <dbl> 5.00, NA, NA, 5.00, 4.91,~
## $ review_scores_location                        <dbl> 4.80, NA, NA, 5.00, 4.94,~
## $ review_scores_value                           <dbl> 5.00, NA, NA, 5.00, 4.92,~
## $ license                                       <chr> NA, NA, NA, NA, NA, NA, N~
## $ instant_bookable                              <lgl> FALSE, FALSE, FALSE, FALS~
## $ calculated_host_listings_count                <dbl> 1, 1, 1, 1, 1, 2, 2, 1, 1~
## $ calculated_host_listings_count_entire_homes   <dbl> 0, 0, 1, 1, 1, 0, 0, 0, 1~
## $ calculated_host_listings_count_private_rooms  <dbl> 1, 1, 0, 0, 0, 2, 2, 1, 0~
## $ calculated_host_listings_count_shared_rooms   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ reviews_per_month                             <dbl> 0.06, NA, NA, 0.03, 1.26,~
```

```
## # A tibble: 1 x 59
##      id source host_id host_name host_since host_location host_response_time
##   <int>  <int>   <int>     <int>      <int>         <int>              <int>
## 1     0      0       0         5          5          8277                  5
## # i 52 more variables: host_response_rate <int>, host_acceptance_rate <int>,
## #   host_is_superhost <int>, host_neighbourhood <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_has_profile_pic <int>, host_identity_verified <int>,
## #   neighbourhood_cleansed <int>, neighbourhood_group_cleansed <int>,
## #   latitude <int>, longitude <int>, property_type <int>, room_type <int>,
## #   accommodates <int>, bathrooms_text <int>, bedrooms <int>, beds <int>, ...
```

70% of the dataset will be reserved for train set and 30% will be reserved for test set.

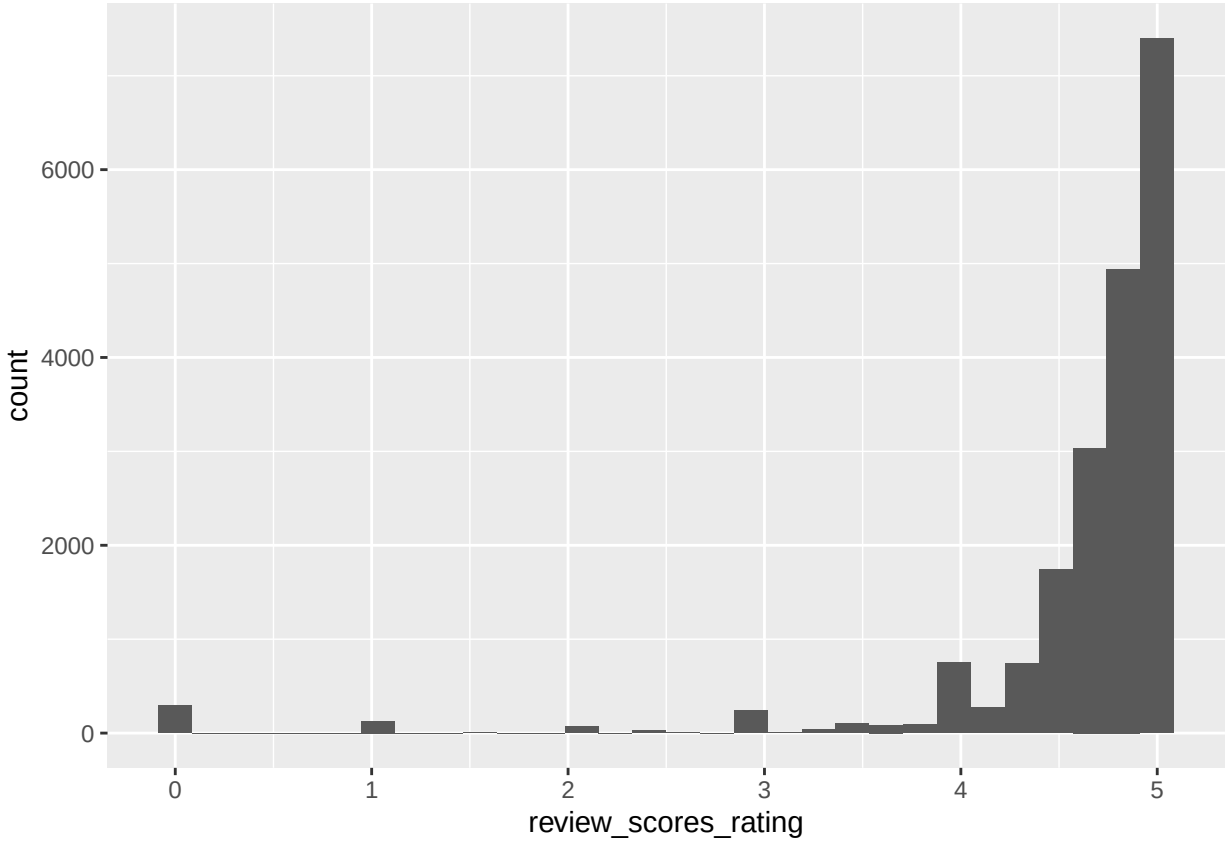
Now we will start transforming the train set. Whatever we do on the train set, we also have to transform the test set.

Let us first Convert total amenities that are in text to total number of amenities in number. This might be a useful transformed column that might affect whether host is super host or not.

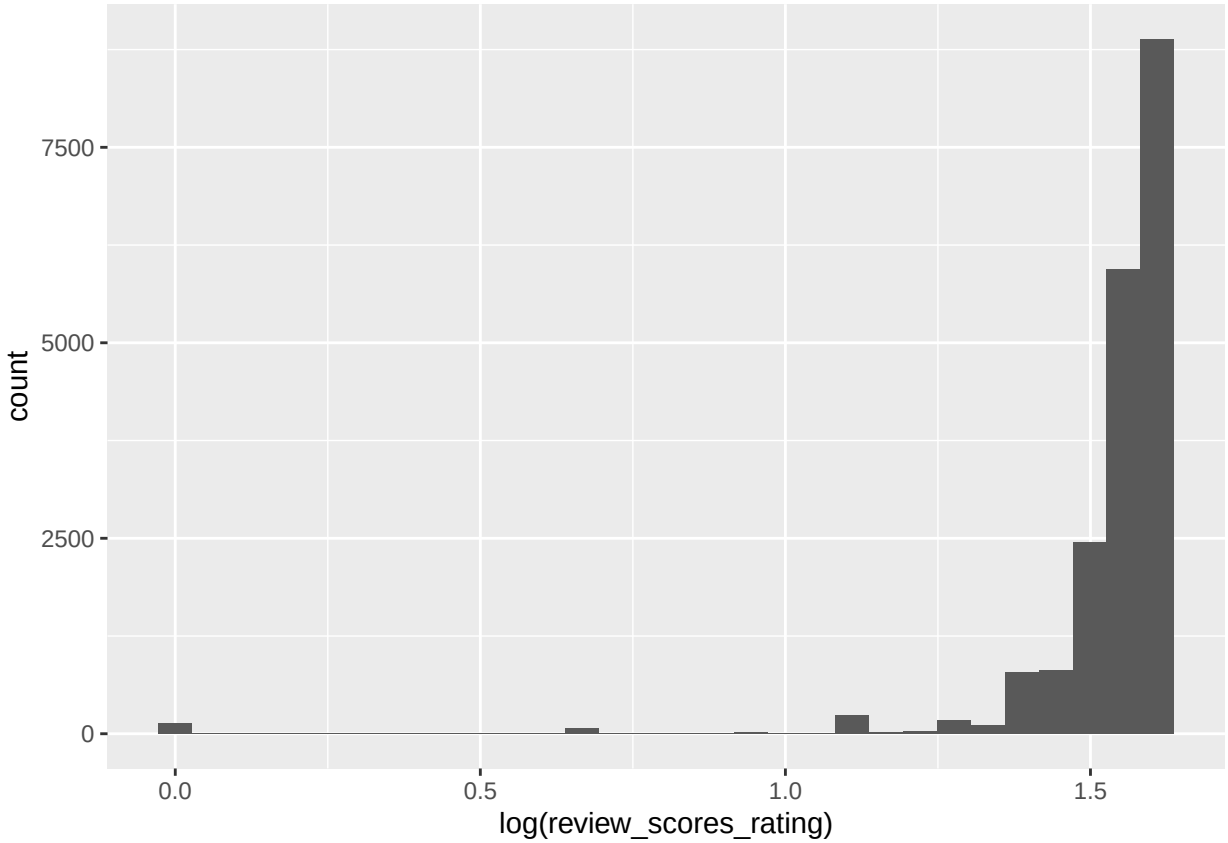
```
## # A tibble: 1 x 60
##      id source host_id host_name host_since host_location host_response_time
##   <int>  <int>   <int>     <int>      <int>         <int>              <int>
## 1     0      0       0         2          2          5782                  2
## # i 53 more variables: host_response_rate <int>, host_acceptance_rate <int>,
```

```
## #   host_is_superhost <int>, host_neighbourhood <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_has_profile_pic <int>, host_identity_verified <int>,
## #   neighbourhood_cleansed <int>, neighbourhood_group_cleansed <int>,
## #   latitude <int>, longitude <int>, property_type <int>, room_type <int>,
## #   accommodates <int>, bathrooms_text <int>, bedrooms <int>, beds <int>, ...
```

There are a lot of null values for quantitative variables i.e review_scores_checkin, review_scores_communication etc. We lose a lot of data if we remove these null value rows. Lets look at the distribution of the values for these variables with removing null values and see if we can replace the null values with the mean or median.



The distribution is left skewed, lets do the log transformation and look at the distribution.

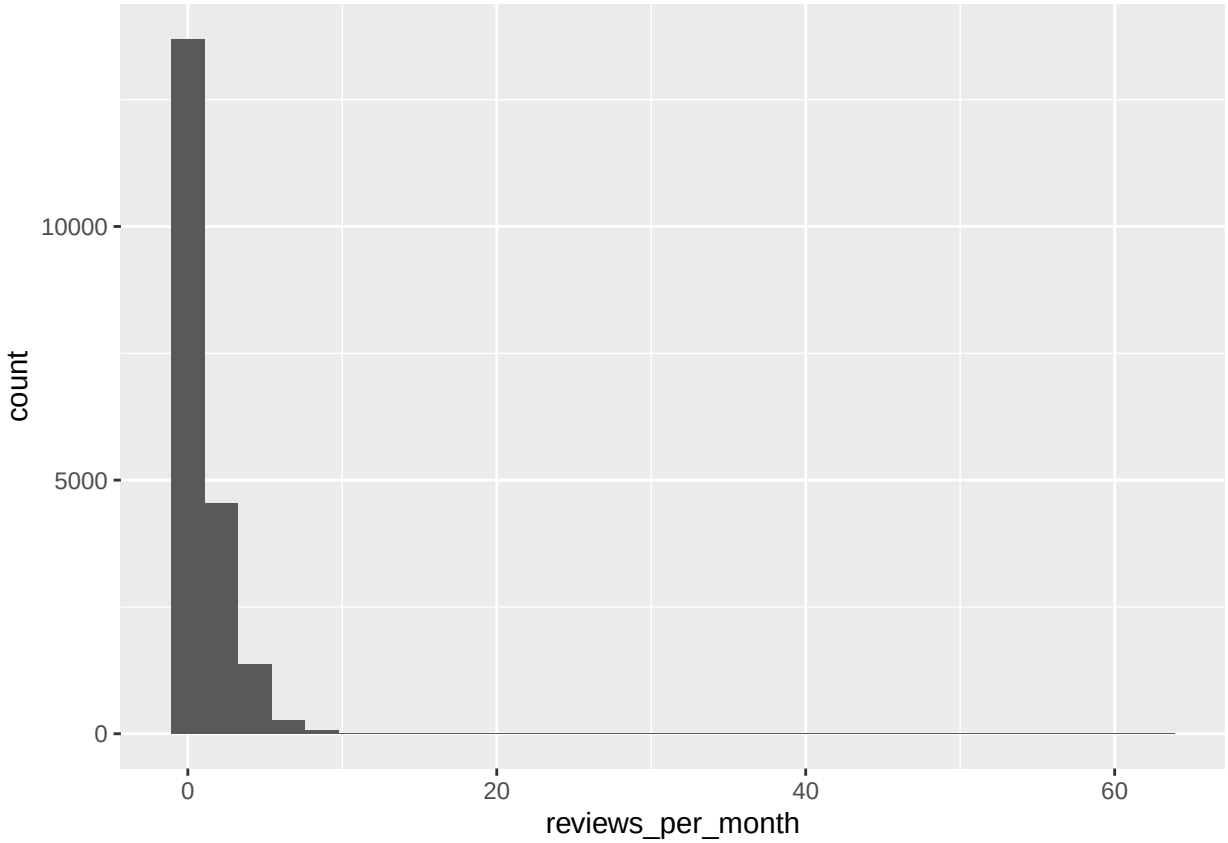


Even the log distribution is left skewed. Lets change the null values of review_scores_rating variable with the median of the distribution as mean will get affected with the skewed distribution. When we transform the data on the training set, we have to note what the median value is on the training set. That will be the value that we will eventually use on our test (instead of converting the NA values on test set with the median of the test set so that we prevent data leakage and have fair evaluation). Both our classification and regression use this column.

Our regression also uses number of bedrooms column and there are some NA values in it. So let us also replace the NA values with the median.

```
## # A tibble: 1 x 60
##      id source host_id host_name host_since host_location host_response_time
##   <int>  <int>   <int>     <int>      <int>         <int>              <int>
## 1     0      0       0         2          2          5782                  2
## # i 53 more variables: host_response_rate <int>, host_acceptance_rate <int>,
## #   host_is_superhost <int>, host_neighbourhood <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_has_profile_pic <int>, host_identity_verified <int>,
## #   neighbourhood_cleansed <int>, neighbourhood_group_cleansed <int>,
## #   latitude <int>, longitude <int>, property_type <int>, room_type <int>,
## #   accommodates <int>, bathrooms_text <int>, bedrooms <int>, beds <int>, ...
```

We have one more variable with a lot of NA values i.e reviews per month. It might be a good predictor variables and might have a impact on the response variables. So lets do some data transformation for that variables since removal of rows will reduce a lot of information in the data.



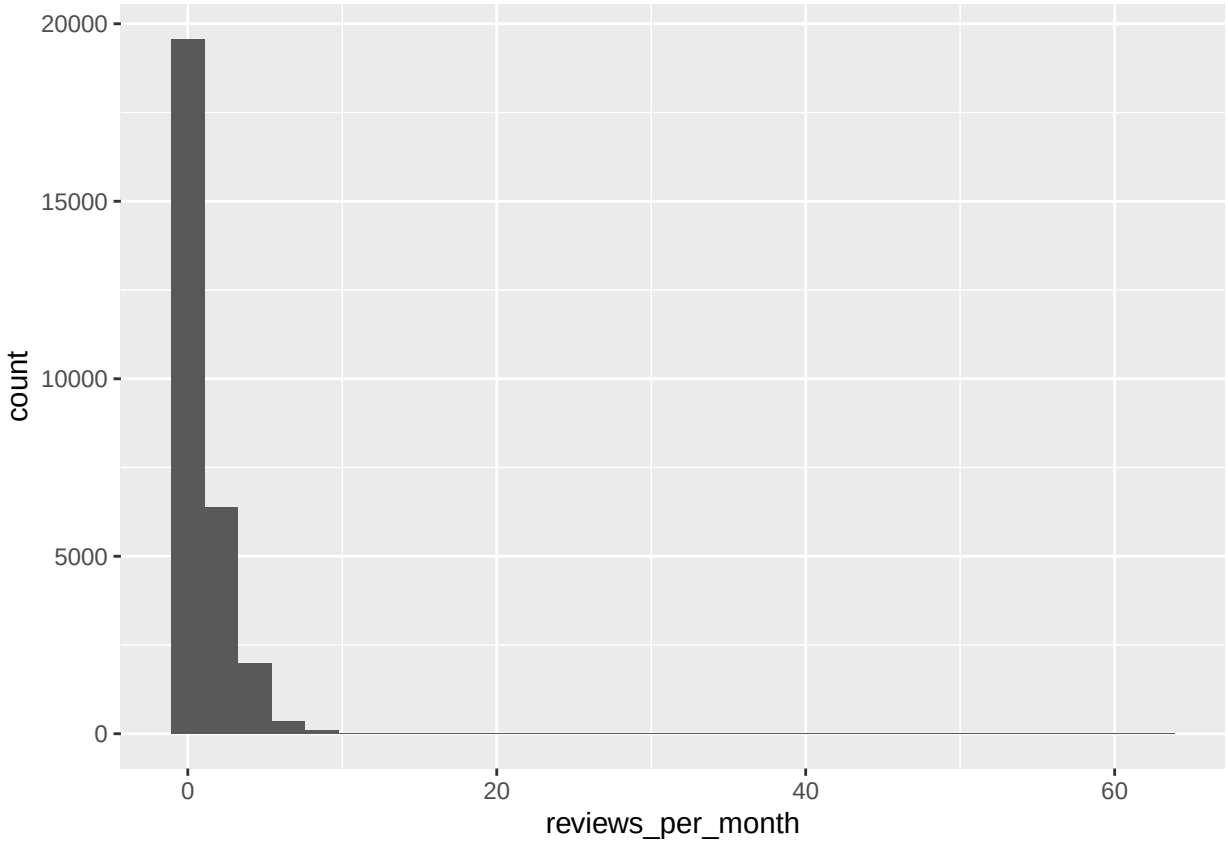
There is a huge outlier in the dataset that have a huge impact on the distribution.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.010   0.110   0.420   1.090   1.542  62.820    7174
```

```
##  [1] 62.82 40.03 39.02 34.89 34.19 33.69 33.45 27.30 25.09 25.00 23.01 16.08
## [13] 15.50 15.14 15.03 15.00 14.94 14.20 13.97 13.97 13.75 13.46 13.00 12.86
## [25] 12.73 12.56 12.03 11.42 11.16 10.75 10.28 10.08  9.96  9.91  9.89  9.86
## [37]  9.82  9.78  9.63  9.54
```

We have around 38K rows and we have few outliers that completely change the shape of the distribution for reviews_per_month variable. Although we can remove outliar s on both trainset and test set, we are not going to do that. We don't have any evidence whether those outliers are because of measurement errors or some sort of error when entered in the dataset. So we are going to keep them. removing them here would also mean removing them from unseen test data. But we still want to show the distribution of the data and see the skew so that we can note that there are outleirs.

The distribution is right skewed. Lets the check the distribution with log transformation.




```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.010   0.110   0.410   1.079   1.520  62.820   10352
```

Replace the null values with the median of the review_per_month distribution.

Remove Na variables from the training set. We will have do this step again on the test set as well. Also remove some columns from the training set such as id, source and amenities. Amenities column have already been transformed to a new column we can drop it here.

```
## # A tibble: 1 x 60
##      id source host_id host_name host_since host_location host_response_time
##   <int>  <int>   <int>     <int>      <int>         <int>              <int>
## 1     0      0       0         2          2          5782                  2
## # i 53 more variables: host_response_rate <int>, host_acceptance_rate <int>,
## #   host_is_superhost <int>, host_neighbourhood <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_has_profile_pic <int>, host_identity_verified <int>,
## #   neighbourhood_cleansed <int>, neighbourhood_group_cleansed <int>,
## #   latitude <int>, longitude <int>, property_type <int>, room_type <int>,
## #   accommodates <int>, bathrooms_text <int>, bedrooms <int>, beds <int>, ...
```

```
## # A tibble: 6 x 57
##     host_id host_name host_since host_location           host_response_time
##       <dbl> <chr>     <chr>      <chr>                   <chr>
## 1  11625123 Jennifer  25/01/14   New York, NY            within an hour
## 2 468732453 Gavriella 10/07/22   <NA>                    within a day
## 3 130314818 Angel     14/05/17   New York, United States within an hour
## 4 215565541 LuxUrban  15/09/18   <NA>                    within an hour
## 5  52577963 Mark      28/12/15   New York, NY            within an hour
```

```
## 6   1403210 Cory      13/11/11   New York, NY            a few days or more
## # i 52 more variables: host_response_rate <chr>, host_acceptance_rate <chr>,
## #   host_is_superhost <lgl>, host_neighbourhood <chr>,
## #   host_listings_count <dbl>, host_total_listings_count <dbl>,
## #   host_has_profile_pic <lgl>, host_identity_verified <lgl>,
## #   neighbourhood_cleansed <chr>, neighbourhood_group_cleansed <chr>,
## #   latitude <dbl>, longitude <dbl>, property_type <chr>, room_type <chr>,
## #   accommodates <dbl>, bathrooms_text <chr>, bedrooms <dbl>, beds <dbl>, ...
```

```
## [1] 8953
```

1.1 Data transformations and EDA

Let us convert the text bath rooms into the numeric columns. Also after doing that let us also remove the NA values from them.

Analyzing the different types of rooms .

```
## [1] "Entire home/apt" "Private room"    "Shared room"     "Hotel room"
```

This is also just looking into the distribution of renters by neighborhood in NewYork.

```
## # A tibble: 5 x 2
##   neighbourhood_group_cleansed count
##   <chr>                        <int>
## 1 Bronx                          913
## 2 Brooklyn                      9667
## 3 Manhattan                    11526
## 4 Queens                        4072
## 5 Staten Island                  273
```

```
## # A tibble: 1 x 58
##   host_id host_name host_since host_location host_response_time
##     <int>     <int>      <int>         <int>              <int>
## 1       0         0          0          5627                  0
## # i 53 more variables: host_response_rate <int>, host_acceptance_rate <int>,
## #   host_is_superhost <int>, host_neighbourhood <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_has_profile_pic <int>, host_identity_verified <int>,
## #   neighbourhood_cleansed <int>, neighbourhood_group_cleansed <int>,
## #   latitude <int>, longitude <int>, property_type <int>, room_type <int>,
## #   accommodates <int>, bathrooms_text <int>, bedrooms <int>, beds <int>, ...
```

So far all these preprocessing steps will be the same for both classification and regression. We will take this train and test data set.Then based on the need of regression and classification we will be selecting columns according to our need.

```
## [1] 26451
```

```
## [1] 11325
```

1.2 1. Classification and regression Models

Creating train and test variable for classification and regression so that both teams can work off separately.

1.3 Classification

1.3.1 Necessary data cleaning, data transformation and data pre-processing.

The price column is supposed to be quantitative variable as it is measured in the number. However in the dataset it is given as charecter as dollar sybmol is used to represent the price. So we transformed the price variable into its ideal type and worked on it. Let us also remove NA values from the price column. This is also our dependent variable for regression and we have to make sure there are no NA values in it.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     8.0    79.0   135.0   216.3   225.0 22286.0
```

```
## # A tibble: 26,451 x 58
##      host_id host_name host_since host_location           host_response_time
##        <dbl> <chr>     <chr>      <chr>                   <chr>
##  1  11625123 Jennifer  25/01/14   New York, NY            within an hour
##  2 468732453 Gavriella 10/07/22   <NA>                    within a day
##  3 130314818 Angel     14/05/17   New York, United States within an hour
##  4 215565541 LuxUrban  15/09/18   <NA>                    within an hour
##  5  52577963 Mark      28/12/15   New York, NY            within an hour
##  6   1403210 Cory      13/11/11   New York, NY            a few days or more
##  7   3801427 Liz       08/10/12   New York, NY            N/A
##  8  50520440 Jessica   04/12/15   New York, NY            N/A
##  9  36166297 Tejal     18/06/15   New York, NY            N/A
## 10 329941458 Gustavo   22/01/20   New York, United States within an hour
## # i 26,441 more rows
## # i 53 more variables: host_response_rate <chr>, host_acceptance_rate <chr>,
## #   host_is_superhost <lgl>, host_neighbourhood <chr>,
## #   host_listings_count <dbl>, host_total_listings_count <dbl>,
## #   host_has_profile_pic <lgl>, host_identity_verified <lgl>,
## #   neighbourhood_cleansed <chr>, neighbourhood_group_cleansed <chr>,
## #   latitude <dbl>, longitude <dbl>, property_type <chr>, room_type <chr>, ...
```

For classification, we can further remove these columns.

There are like 5-6 variables related to review scores and in overall all are related to each other. So let us only keep the review score rating and get rid of other variables like review_scores_accuracy, review_scores_cleanliness, review_scores_checkin etc.

```
## # A tibble: 1 x 45
##   host_id host_name host_since host_response_time host_response_rate
##     <int>     <int>      <int>              <int>              <int>
## 1       0         0          0                  0                  0
## # i 40 more variables: host_acceptance_rate <int>, host_is_superhost <int>,
## #   host_neighbourhood <int>, host_listings_count <int>,
## #   host_total_listings_count <int>, host_has_profile_pic <int>,
## #   host_identity_verified <int>, neighbourhood_cleansed <int>,
## #   neighbourhood_group_cleansed <int>, latitude <int>, longitude <int>,
## #   property_type <int>, room_type <int>, accommodates <int>,
## #   bathrooms_text <int>, bedrooms <int>, beds <int>, price <int>, ...
```

1.4 Introduction and Background:

Customer satisfaction is one of the primary objective that every business owner will focus on. Providing them with the best of what they asked for is very important to run a successful business.This is where a strong trust will establish between customers and business domain .In this project we will be focusing on airbnb data with main idea to analyze the historical data we have and find pattern in the data that can help improve the customer satisfaction and help the company increase their business.

Let us have a look at the dataset and see what all variables we have:

```
## Rows: 26,451
## Columns: 45
## $ host_id                                       <dbl> 11625123, 468732453, 1303~
## $ host_name                                     <chr> "Jennifer", "Gavriella", ~
## $ host_since                                    <chr> "25/01/14", "10/07/22", "~
## $ host_response_time                            <chr> "within an hour", "within~
## $ host_response_rate                            <chr> "100%", "90%", "100%", "9~
## $ host_acceptance_rate                          <chr> "100%", "80%", "98%", "98~
## $ host_is_superhost                             <lgl> FALSE, TRUE, FALSE, FALSE~
## $ host_neighbourhood                            <chr> "Crown Heights", "Crown H~
## $ host_listings_count                           <dbl> 2, 3, 5, 60, 5, 1, 1, 1, ~
## $ host_total_listings_count                     <dbl> 2, 4, 5, 93, 7, 2, 2, 2, ~
## $ host_has_profile_pic                          <lgl> TRUE, TRUE, TRUE, TRUE, T~
## $ host_identity_verified                        <lgl> TRUE, FALSE, TRUE, TRUE, ~
## $ neighbourhood_cleansed                        <chr> "Crown Heights", "Crown H~
## $ neighbourhood_group_cleansed                  <chr> "Brooklyn", "Brooklyn", "~
## $ latitude                                      <dbl> 40.67630, 40.66423, 40.76~
## $ longitude                                     <dbl> -73.95903, -73.93151, -73~
## $ property_type                                 <chr> "Entire rental unit", "En~
## $ room_type                                     <chr> "Entire home/apt", "Entir~
## $ accommodates                                  <dbl> 4, 9, 2, 2, 2, 6, 3, 2, 1~
## $ bathrooms_text                                <chr> "1 bath", "2 baths", "1 s~
## $ bedrooms                                      <dbl> 1, 3, 1, 1, 1, 2, 2, 1, 1~
## $ beds                                          <dbl> 2, 8, 1, 1, 1, 3, 2, 1, 1~
## $ price                                         <dbl> 95, 231, 140, 771, 99, 21~
## $ minimum_nights                                <dbl> 30, 30, 30, 1, 5, 30, 30,~
## $ maximum_nights                                <dbl> 1125, 40, 40, 365, 29, 11~
## $ minimum_nights_avg_ntm                        <dbl> 30.0, 30.0, 30.0, 1.0, 5.~
## $ maximum_nights_avg_ntm                        <dbl> 1125, 40, 40, 365, 29, 11~
## $ has_availability                              <lgl> TRUE, TRUE, TRUE, TRUE, T~
## $ availability_30                               <dbl> 0, 7, 26, 0, 8, 28, 0, 0,~
## $ availability_60                               <dbl> 0, 28, 56, 0, 18, 58, 0, ~
## $ availability_90                               <dbl> 0, 58, 86, 0, 30, 88, 0, ~
## $ availability_365                              <dbl> 0, 224, 266, 0, 284, 363,~
## $ calendar_last_scraped                         <chr> "02/10/23", "01/10/23", "~
## $ number_of_reviews                             <dbl> 24, 16, 1, 1, 39, 7, 27, ~
## $ number_of_reviews_ltm                         <dbl> 0, 16, 1, 1, 13, 0, 0, 0,~
## $ number_of_reviews_l30d                        <dbl> 0, 3, 1, 0, 2, 0, 0, 0, 0~
## $ review_scores_rating                          <dbl> 4.75, 4.94, 5.00, 4.00, 4~
## $ instant_bookable                              <lgl> TRUE, FALSE, FALSE, TRUE,~
## $ calculated_host_listings_count                <dbl> 2, 3, 5, 36, 2, 1, 1, 1, ~
## $ calculated_host_listings_count_entire_homes   <dbl> 2, 1, 0, 0, 1, 1, 1, 0, 0~
## $ calculated_host_listings_count_private_rooms  <dbl> 0, 2, 5, 36, 1, 0, 0, 1, ~
## $ calculated_host_listings_count_shared_rooms   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ reviews_per_month                             <dbl> 0.25, 2.91, 1.00, 0.54, 1~
```

```
## $ amenities_count                          <dbl> 18, 42, 39, 4, 38, 19, 40~
## $ bathroom                                 <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, ~
```

The dataset contains information about the host who has a listing in airbnb, information/specifications about the listings. Here we can improve the customers satisfaction by providing them a measure that assist them to get the best stay with the all the possibilities they have In the data set we have a variable called "host is superhost" which is a Boolean variable which says if the host, who has his listings in airbnb, is given a badge called superhost by the airbnb management.

This is how the Airbnb management provide the superhost badge to the hosts: They say a Superhost is a Host who goes above and beyond to provide excellent hospitality. The guests can easily identify a Superhost from the badge that appears on their Airbnb listing and profile. These are the requirement the host should have that are set by Airbnb to receive a badge a super host: - completed at least 10 trips or 3 reservations that total at least 100 nights - Maintained a 90% response rate or higher - Maintained a less than 1% cancellation rate, with exceptions made for those that fall under our Extenuating Circumstances policy - Maintained a 4.8 overall rating (A review counts towards superhost status when either both the guest and the Host have submitted a review, or the 14-day window for reviews is over, whichever comes first).

These are the 4 main characteristics the airbnb focus on to provide a superhost badge. However there are many variables in the dataset that might have different patterns in the data for both superhost and not superhost.

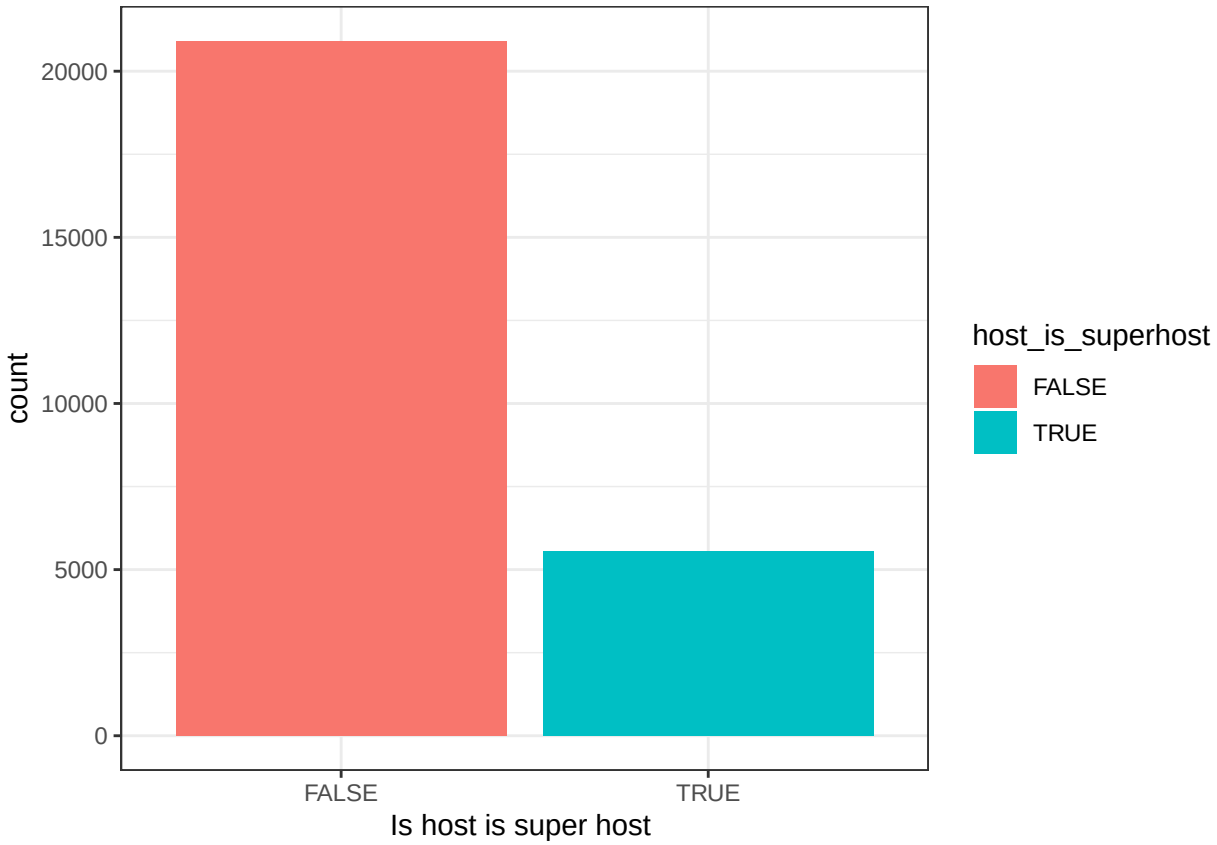
So the main idea is to create a classification task to classify if the host is superhost with considering the predictor variables that has a good impact and adds weightage to predict if the host is superhost. This will helps the airbnb company to make more legitimate decisions if the host is eligible for receiving the superhost badge without only considering their specific requirements but also with all the data they have regarding the host as well as listings. They will also gain more trust with their customers and also help him to make better decision with the host is superhost variable.

1.5 Data pre-processing and EDA:

Before we fit any models we have to make sure that the data is clean, transformed into its original form and ready to fit the machine learning models.

Lets first look at the distribution of in host_is_superhost column

```
##
## FALSE  TRUE
## 20911  5540
```



Around 20 to 25 % seem to be super host which is reasonable.

```
## # A tibble: 1 x 45
##   host_id host_name host_since host_response_time host_response_rate
##     <int>     <int>      <int>              <int>              <int>
## 1       0         0          0                  0                  0
## # i 40 more variables: host_acceptance_rate <int>, host_is_superhost <int>,
## #   host_neighbourhood <int>, host_listings_count <int>,
## #   host_total_listings_count <int>, host_has_profile_pic <int>,
## #   host_identity_verified <int>, neighbourhood_cleansed <int>,
## #   neighbourhood_group_cleansed <int>, latitude <int>, longitude <int>,
## #   property_type <int>, room_type <int>, accommodates <int>,
## #   bathrooms_text <int>, bedrooms <int>, beds <int>, price <int>, ...
```

- It appears there are still null values in a host_neighbourhood column and it is 20% of the complete train data. Removing all those will result in data truncation. Host neighborhood contains the neighborhood counties of the host and we think it will effect the superhost decision. So we get rid of the column data.

```
##
## FALSE  TRUE
## 20911  5540
```

For the variables host_reponse_time and host_acceptance_rate which is of character type also has NA values in it. We cannot remove those columns as they will have significant impact on the response variable. so simply lets get rid of NA values for these columns in both train and test set.

1.6 Type Conversion:

There are 4 important variables that shows about the characteristics of the host and are very essential for predicting host is superhost that are in wrong data type. So we have to convert it into its original type for both train and test

data.

```
## # A tibble: 1 x 44
##   host_id host_name host_since host_response_time host_response_rate
##     <int>     <int>      <int>              <int>              <int>
## 1       0         0          0                  0                  0
## # i 39 more variables: host_acceptance_rate <int>, host_is_superhost <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_has_profile_pic <int>, host_identity_verified <int>,
## #   neighbourhood_cleansed <int>, neighbourhood_group_cleansed <int>,
## #   latitude <int>, longitude <int>, property_type <int>, room_type <int>,
## #   accommodates <int>, bathrooms_text <int>, bedrooms <int>, beds <int>,
## #   price <int>, minimum_nights <int>, maximum_nights <int>, ...
```

```
## # A tibble: 6 x 44
##     host_id host_name host_since host_response_time host_response_rate
##       <dbl> <chr>     <date>                 <int>              <dbl>
## 1  11625123 Jennifer  2014-01-25                 4                100
## 2 468732453 Gavriella 2022-07-10                 2                 90
## 3 130314818 Angel     2017-05-14                 4                100
## 4 215565541 LuxUrban  2018-09-15                 4                 96
## 5  52577963 Mark      2015-12-28                 4                100
## 6   1403210 Cory      2011-11-13                 1                  0
## # i 39 more variables: host_acceptance_rate <dbl>, host_is_superhost <lgl>,
## #   host_listings_count <dbl>, host_total_listings_count <dbl>,
## #   host_has_profile_pic <lgl>, host_identity_verified <lgl>,
## #   neighbourhood_cleansed <chr>, neighbourhood_group_cleansed <chr>,
## #   latitude <dbl>, longitude <dbl>, property_type <chr>, room_type <chr>,
## #   accommodates <dbl>, bathrooms_text <chr>, bedrooms <dbl>, beds <dbl>,
## #   price <dbl>, minimum_nights <dbl>, maximum_nights <dbl>, ...
```

Finally lets look the distribution of levels in superhost.

```
##
## FALSE  TRUE
## 11852  5342
```

Now we are good with the NA values and other data transformation that are essential for model fiting. However we have few variables in the dataset that doesn't make such sense or adds weightage to predict host is super host like host_id, host_name, neighbourhood_cleansed, neighbourhood_group_cleansed etc. So we get rid of all those and only consider variables that displays the charecteristics of the host and his listings.

```
## # A tibble: 6 x 22
##   host_since host_response_time host_response_rate host_acceptance_rate
##   <date>                 <int>              <dbl>                <dbl>
## 1 2014-01-25                 4                100                  100
## 2 2022-07-10                 2                 90                   80
## 3 2017-05-14                 4                100                   98
## 4 2018-09-15                 4                 96                   98
## 5 2015-12-28                 4                100                   96
## 6 2011-11-13                 1                  0                    0
## # i 18 more variables: host_is_superhost <lgl>,
## #   host_total_listings_count <dbl>, host_has_profile_pic <lgl>,
```

```
## #   host_identity_verified <lgl>, calculated_host_listings_count <dbl>,
## #   calculated_host_listings_count_entire_homes <dbl>,
## #   calculated_host_listings_count_private_rooms <dbl>,
## #   calculated_host_listings_count_shared_rooms <dbl>, instant_bookable <lgl>,
## #   availability_30 <dbl>, price <dbl>, amenities_count <dbl>, ...
```

```
## [1] 17194    22
```

```
## [1] 7320   22
```

The dataset is now clean with around 17194 observations (for training set) and 4423 for test set and 22 variables are used to uncover the different characteristics of the host to classify whether the host is considered super host or not.

1.7 Modeling

```
##
## FALSE  TRUE
##  5089  2231
```

```
## # A tibble: 6 x 22
##   host_since host_response_time host_response_rate host_acceptance_rate
##   <date>                 <int>              <dbl>                <dbl>
## 1 2014-01-25                 4                100                  100
## 2 2022-07-10                 2                 90                   80
## 3 2017-05-14                 4                100                   98
## 4 2018-09-15                 4                 96                   98
## 5 2015-12-28                 4                100                   96
## 6 2011-11-13                 1                  0                    0
## # i 18 more variables: host_is_superhost <lgl>,
## #   host_total_listings_count <dbl>, host_has_profile_pic <lgl>,
## #   host_identity_verified <lgl>, calculated_host_listings_count <dbl>,
## #   calculated_host_listings_count_entire_homes <dbl>,
## #   calculated_host_listings_count_private_rooms <dbl>,
## #   calculated_host_listings_count_shared_rooms <dbl>, instant_bookable <lgl>,
## #   availability_30 <dbl>, price <dbl>, amenities_count <dbl>, ...
```

```
##
## Call:
## glm(formula = as.factor(host_is_superhost) ~ ., family = "binomial",
##     data = train_cf)
##
## Coefficients:
##                                          Estimate Std. Error z value
## (Intercept)                            -2.051e+01  7.396e-01 -27.732
## host_since                             -1.499e-04  1.637e-05  -9.152
## host_response_time                      1.277e-01  3.569e-02   3.577
## host_response_rate                      2.955e-02  2.649e-03  11.156
## host_acceptance_rate                    2.152e-02  1.264e-03  17.025
## host_total_listings_count              -1.084e-03  1.841e-04  -5.889
## host_has_profile_picTRUE                1.170e+00  1.858e-01   6.295
## host_identity_verifiedTRUE             -8.620e-02  6.829e-02  -1.262
## calculated_host_listings_count         -3.823e-01  1.162e-01  -3.290
```

```
## calculated_host_listings_count_entire_homes   3.627e-01  1.163e-01   3.120
## calculated_host_listings_count_private_rooms  3.792e-01  1.162e-01   3.263
## calculated_host_listings_count_shared_rooms  -3.449e-02  1.421e-01  -0.243
## instant_bookableTRUE                         -4.287e-01  4.986e-02  -8.598
## availability_30                              -6.374e-03  1.841e-03  -3.462
## price                                        -2.514e-04  9.309e-05  -2.701
## amenities_count                               2.452e-02  1.389e-03  17.651
## bathroom                                     -2.137e-02  4.462e-02  -0.479
## accommodates                                 -2.830e-02  1.561e-02  -1.813
## beds                                          2.164e-02  2.499e-02   0.866
## minimum_nights                                9.759e-04  1.029e-03   0.949
## review_scores_rating                          3.270e+00  1.225e-01  26.704
## reviews_per_month                             1.430e-01  1.309e-02  10.925
##                                              Pr(>|z|)
## (Intercept)                                   < 2e-16 ***
## host_since                                    < 2e-16 ***
## host_response_time                           0.000348 ***
## host_response_rate                            < 2e-16 ***
## host_acceptance_rate                          < 2e-16 ***
## host_total_listings_count                    3.88e-09 ***
## host_has_profile_picTRUE                     3.07e-10 ***
## host_identity_verifiedTRUE                   0.206819
## calculated_host_listings_count               0.001001 **
## calculated_host_listings_count_entire_homes  0.001810 **
## calculated_host_listings_count_private_rooms 0.001102 **
## calculated_host_listings_count_shared_rooms  0.808199
## instant_bookableTRUE                          < 2e-16 ***
## availability_30                              0.000536 ***
## price                                        0.006922 **
## amenities_count                               < 2e-16 ***
## bathroom                                     0.631940
## accommodates                                 0.069806 .
## beds                                         0.386540
## minimum_nights                               0.342689
## review_scores_rating                          < 2e-16 ***
## reviews_per_month                             < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 21309  on 17193  degrees of freedom
## Residual deviance: 15316  on 17172  degrees of freedom
## AIC: 15360
##
## Number of Fisher Scoring iterations: 8
```

1.8 Find the most significant predictor variables from the logistic regression and use the reduced model for analysis.

```
##
## Call:
## glm(formula = as.factor(host_is_superhost) ~ host_since + host_acceptance_rate +
##     host_response_time + review_scores_rating + reviews_per_month +
```

```
##     host_total_listings_count + host_has_profile_pic + host_response_rate +
##     calculated_host_listings_count_entire_homes + calculated_host_listings_count_private_rooms +
##     instant_bookable + price + availability_30 + amenities_count,
##     family = "binomial", data = train_cf)
##
## Coefficients:
##                                             Estimate Std. Error z value
## (Intercept)                                -2.077e+01  7.333e-01 -28.327
## host_since                                 -1.501e-04  1.625e-05  -9.238
## host_acceptance_rate                        2.126e-02  1.259e-03  16.880
## host_response_time                          1.198e-01  3.552e-02   3.371
## review_scores_rating                        3.303e+00  1.220e-01  27.066
## reviews_per_month                           1.421e-01  1.285e-02  11.060
## host_total_listings_count                  -1.062e-03  1.823e-04  -5.827
## host_has_profile_picTRUE                    1.148e+00  1.856e-01   6.187
## host_response_rate                          3.023e-02  2.653e-03  11.393
## calculated_host_listings_count_entire_homes -2.018e-02  1.817e-03 -11.109
## calculated_host_listings_count_private_rooms -3.111e-03  6.083e-04  -5.115
## instant_bookableTRUE                       -4.468e-01  4.959e-02  -9.010
## price                                      -3.641e-04  8.711e-05  -4.181
## availability_30                            -6.899e-03  1.830e-03  -3.771
## amenities_count                             2.464e-02  1.369e-03  18.001
##                                            Pr(>|z|)
## (Intercept)                                 < 2e-16 ***
## host_since                                  < 2e-16 ***
## host_acceptance_rate                        < 2e-16 ***
## host_response_time                         0.000748 ***
## review_scores_rating                        < 2e-16 ***
## reviews_per_month                           < 2e-16 ***
## host_total_listings_count                 5.64e-09 ***
## host_has_profile_picTRUE                  6.12e-10 ***
## host_response_rate                          < 2e-16 ***
## calculated_host_listings_count_entire_homes  < 2e-16 ***
## calculated_host_listings_count_private_rooms 3.13e-07 ***
## instant_bookableTRUE                        < 2e-16 ***
## price                                      2.91e-05 ***
## availability_30                            0.000163 ***
## amenities_count                             < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 21309  on 17193  degrees of freedom
## Residual deviance: 15380  on 17179  degrees of freedom
## AIC: 15410
##
## Number of Fisher Scoring iterations: 8


## Call:
## lda(as.factor(host_is_superhost) ~ host_since + host_acceptance_rate +
##     host_response_time + review_scores_rating + reviews_per_month +
##     host_total_listings_count + host_has_profile_pic + host_response_rate +
##     calculated_host_listings_count_entire_homes + price + calculated_host_listings_count_private_ro
```

```
##     instant_bookable + availability_30 + amenities_count, data = train_cf)
##
## Prior probabilities of groups:
##     FALSE      TRUE
## 0.6893102 0.3106898
##
## Group means:
##       host_since host_acceptance_rate host_response_time review_scores_rating
## FALSE   17445.50             78.16639           3.366689             4.675472
## TRUE    17224.36             91.10558           3.671097             4.856849
##       reviews_per_month host_total_listings_count host_has_profile_picTRUE
## FALSE         0.9671861                 505.82703                0.9798346
## TRUE          1.8158368                  19.00337                0.9923250
##       host_response_rate calculated_host_listings_count_entire_homes    price
## FALSE           90.83834                                    38.67575 244.6504
## TRUE            97.89835                                     2.91052 203.1469
##       calculated_host_listings_count_private_rooms instant_bookableTRUE
## FALSE                                    44.121498            0.2667904
## TRUE                                      5.475103            0.2072258
##       availability_30 amenities_count
## FALSE       10.267550        27.57568
## TRUE         9.387495        38.65144
##
## Coefficients of linear discriminants:
##                                                    LD1
## host_since                                -0.0001472531
## host_acceptance_rate                       0.0137815430
## host_response_time                         0.0950524153
## review_scores_rating                       0.8386828798
## reviews_per_month                          0.1603062468
## host_total_listings_count                 -0.0001023646
## host_has_profile_picTRUE                   0.6332354430
## host_response_rate                         0.0093417618
## calculated_host_listings_count_entire_homes  -0.0031539087
## price                                     -0.0001258830
## calculated_host_listings_count_private_rooms -0.0015988423
## instant_bookableTRUE                      -0.6100864278
## availability_30                           -0.0066915164
## amenities_count                            0.0292406937


## Call:
## qda(as.factor(host_is_superhost) ~ host_since + host_acceptance_rate +
##     +host_response_time + review_scores_rating + reviews_per_month +
##     host_total_listings_count + host_has_profile_pic + host_response_rate +
##     calculated_host_listings_count_entire_homes + price + calculated_host_listings_count_private_ro
##     instant_bookable + availability_30 + amenities_count, data = trainData)
##
## Prior probabilities of groups:
##     FALSE      TRUE
## 0.6893102 0.3106898
##
## Group means:
##       host_since host_acceptance_rate host_response_time review_scores_rating
## FALSE   17445.50             78.16639           3.366689             4.675472
```

```
## TRUE    17224.36             91.10558           3.671097             4.856849
##       reviews_per_month host_total_listings_count host_has_profile_picTRUE
## FALSE         0.9671861                 505.82703                0.9798346
## TRUE          1.8158368                  19.00337                0.9923250
##       host_response_rate calculated_host_listings_count_entire_homes    price
## FALSE           90.83834                                     38.67575 244.6504
## TRUE            97.89835                                      2.91052 203.1469
##       calculated_host_listings_count_private_rooms instant_bookableTRUE
## FALSE                                    44.121498            0.2667904
## TRUE                                      5.475103            0.2072258
##       availability_30 amenities_count
## FALSE       10.267550        27.57568
## TRUE         9.387495        38.65144
```

1.9 Summarize your findings in terms of the correct classification rate.

```
## [1] 0.7821038 0.7609290 0.6221311
```

LDA and Logistic regression appear to have higer accuracy on the test set. This is an indication that the data might be linearly separable

1.10 Let's now explore KNN as well with the predictors from the reduced model to identify the best k and its correct classification rate.

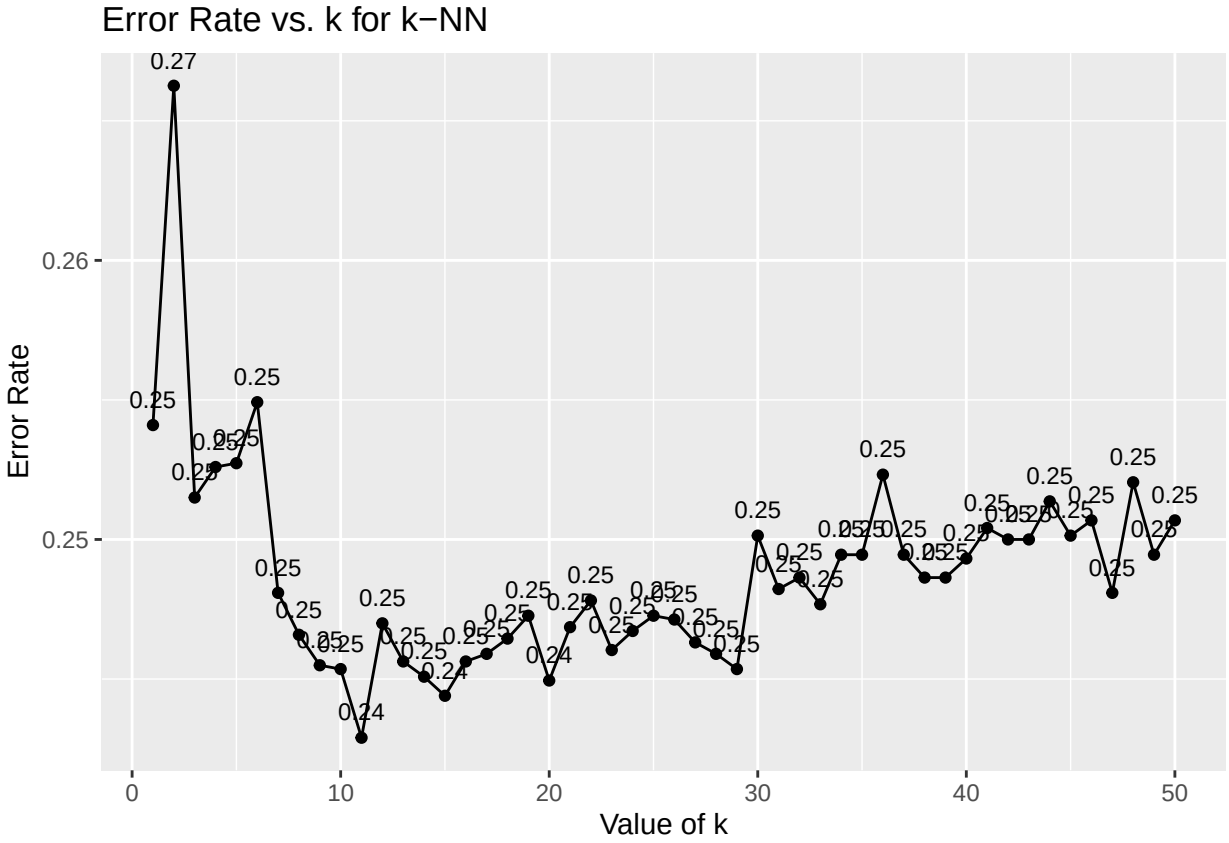
```
## # A tibble: 6 x 13
##   host_acceptance_rate host_total_listings_count host_has_profile_pic
##                  <dbl>                     <dbl> <lgl>
## 1                  100                         2 TRUE
## 2                   80                         4 TRUE
## 3                   98                         5 TRUE
## 4                   98                        93 TRUE
## 5                   96                         7 TRUE
## 6                    0                         2 TRUE
## # i 10 more variables: host_response_rate <dbl>,
## #   calculated_host_listings_count_entire_homes <dbl>, price <dbl>,
## #   calculated_host_listings_count_private_rooms <dbl>, instant_bookable <lgl>,
## #   availability_30 <dbl>, amenities_count <dbl>, host_response_time <dbl>,
## #   review_scores_rating <dbl>, reviews_per_month <dbl>

## # A tibble: 1 x 13
##   host_acceptance_rate host_total_listings_count host_has_profile_pic
##                  <int>                     <int>                <int>
## 1                    0                         0                    0
## # i 10 more variables: host_response_rate <int>,
## #   calculated_host_listings_count_entire_homes <int>, price <int>,
## #   calculated_host_listings_count_private_rooms <int>, instant_bookable <int>,
## #   availability_30 <int>, amenities_count <int>, host_response_time <int>,
## #   review_scores_rating <int>, reviews_per_month <int>

## [1] 7320   13
```

Let's fine tune the best k parameter with training model on different K's on the training dataset and choose k that has less error rate on test dataset. Let's choose a k value from 1 to 50

```
## [1] 0.2428962
```

1.11 Let's explore KNN as to identify the best k and its correct classification rate.

```
## [1] "The best k is  11  with an error rate of  24.3 %"
```



1.12 Observation

The best k is 11 with the lowest error rate

```
## [1] 0.7568306
```

1.12.1 We have Knn accuracy with the best k still performing lower than both logistic and QDA

- Let's now explore Support Vector Machine on our dataset

```
##
## Yhat_all FALSE TRUE
##    FALSE  4390  908
##    TRUE    699 1323
```

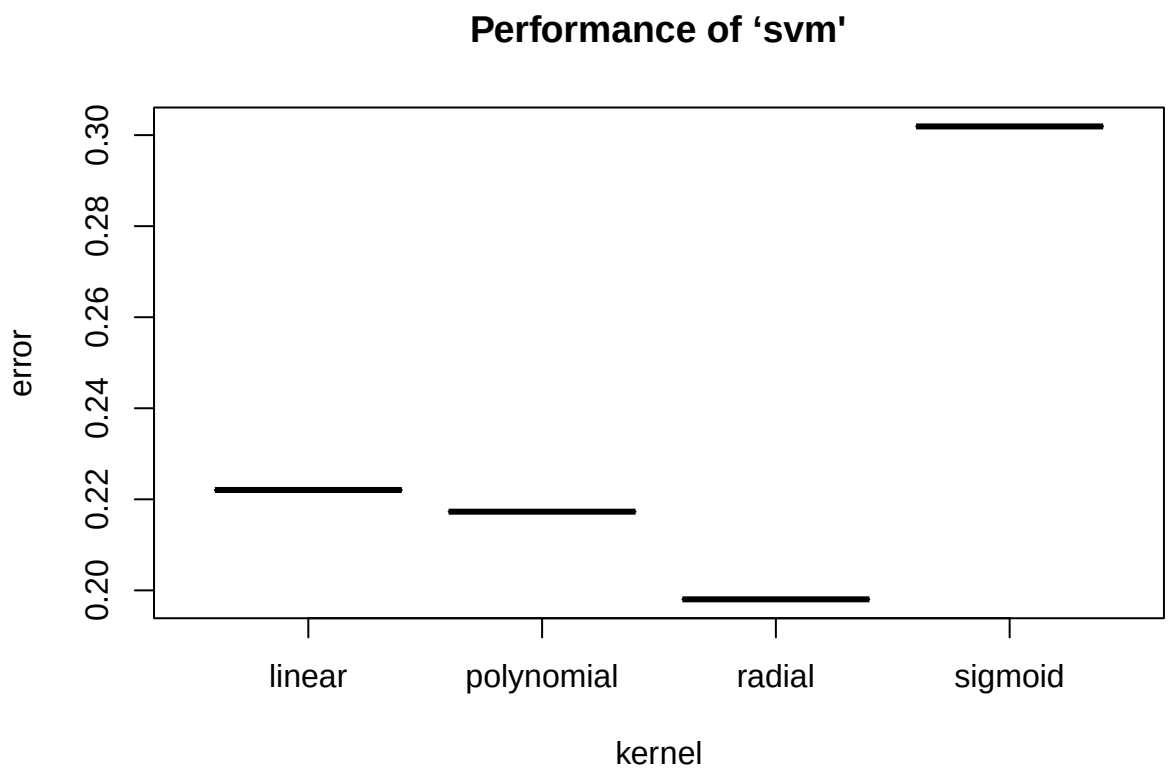
```
## [1] 0.7804645
```

1.13 With SVM, we got about 78% on the test

- Let's tune the hyperparameters - kernel and the cost budget

```
## [1] 0.8019669

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  kernel
##  radial
##
## - best performance: 0.1980331
##
## - Detailed performance results:
##       kernel     error  dispersion
## 1     linear 0.2220536 0.008668506
## 2 polynomial 0.2172848 0.004075654
## 3     radial 0.1980331 0.006245381
## 4    sigmoid 0.3019073 0.016096372
```

1.14 It looks like radial kernel is the best for this dataset



1.15 Radial kernel has the least error as seen in the plot.

Lets now tune SVM based on Cost Budget.

Note we can tune svm based on both kernel and cost budget and select the best combination of the two but that's computation intensive. So I am going to select the radial kernel gotten above and just tune cost budget with it.

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost kernel
##   2.4 radial
##
## - best performance: 0.06058941
##
## - Detailed performance results:
##    cost kernel      error dispersion
## 1   0.1 radial 0.08116883 0.04627136
## 2   0.2 radial 0.07217782 0.03814516
## 3   0.3 radial 0.07474192 0.03543490
## 4   0.4 radial 0.07345987 0.03453698
## 5   0.5 radial 0.06958042 0.03236100
## 6   0.6 radial 0.06958042 0.03236100
## 7   0.7 radial 0.06573427 0.02898621
## 8   0.8 radial 0.06573427 0.02898621
## 9   0.9 radial 0.06573427 0.02898621
## 10  1.0 radial 0.06829837 0.02602873
## 11  1.1 radial 0.06958042 0.02467643
## 12  1.2 radial 0.06443556 0.02734751
## 13  1.3 radial 0.06573427 0.02564316
## 14  1.4 radial 0.06573427 0.02564316
## 15  1.5 radial 0.06573427 0.02564316
## 16  1.6 radial 0.06443556 0.02528367
## 17  1.7 radial 0.06443556 0.02528367
## 18  1.8 radial 0.06187146 0.02792775
## 19  1.9 radial 0.06187146 0.02792775
## 20  2.0 radial 0.06317016 0.02838427
## 21  2.1 radial 0.06058941 0.02606736
## 22  2.2 radial 0.06058941 0.02606736
## 23  2.3 radial 0.06187146 0.02587329
## 24  2.4 radial 0.06058941 0.02809064
## 25  2.5 radial 0.06058941 0.02809064
## 26  2.6 radial 0.06058941 0.02809064
## 27  2.7 radial 0.06188811 0.02925701
## 28  2.8 radial 0.06188811 0.02925701
## 29  2.9 radial 0.06058941 0.02809064
## 30  3.0 radial 0.06058941 0.02809064
```

1.16 Best parameters are cost = 2.4, and kernel = radial

- Let's create the final Best model.

```
##
```

```
## Yhat_all FALSE TRUE
##    FALSE  4544  834
##    TRUE    545 1397
```

```
## [1] 0.811612
```

1.17 Incredible! With SVM, we reached a whopping 81% accuracy on the test set. That's so impressive for this task.

1.18 Final Note.

```
SVM is one of the most power machine learning algorithm that exists today. It is effective in
high dimensional spaces, works well with a clear margin of separation but it require a robust
hyperparameter tuning. Careful tuning can help with selection of the right kernel and cost
budget for the data.
```

1.19 Regression

This data set contains information about prices, location, reviews, room types, host and more for over 30000 room listings. On this project, we will be using this data set to train a machine learning algorithm in order to do regression on prices.
The data contains over 75 columns and 38,792 rows/observations. We will only choose certain number of columns as predictor variables. We choose price to be our dependent variable and 14 columns/variables as our independent variables as listed below. . Since, some of this variables are catagorical variable the total number of predictors will be eventually be close to 20 since many algorithms might hot encode or add dummy varibales for catogorical features under the hood

Dependent variable: price.

Independent Variables.

room_type.
neighbourhood_group_cleansed.
instant_bookable.
bedrooms.
minimum_nights. maximum_nights. latitude. longitude.
availability_30 accommodates.
host_listings_count.
host_total_listings_count.
host_is_superhost.
review_scores_rating.

2 1. Loading the Data.

```
## function (x, df1, df2, ncp, log = FALSE)
```

3 2. Data Transformation

Creating Training and testing data. We wtook 70% of the data for the training and 30% of the data for testing previosuly.

Filter the columns that we need from the Dataset for regression.

Count the NAS for each column on the training set. There are some columns that have a lot of NA values in them. For example bedrooms and review_scores_rating have over 7000 NA observations in them. Instead of removing those observations we will be replace the NA values with the median of their respective columns. Since that is a lot of observations (around 25% of the data) we didnt want to remove them. We felt replacing the values with the median will do.

```
## # A tibble: 1 x 15
##   price room_type neighbourhood_group_cleansed instant_bookable bedrooms
##   <int>     <int>                         <int>            <int>    <int>
## 1     0         0                             0                0        0
## # i 10 more variables: minimum_nights <int>, maximum_nights <int>,
## #   latitude <int>, longitude <int>, availability_30 <int>, accommodates <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_is_superhost <int>, review_scores_rating <int>
```

Replace with median for reviewScore and number of bedrooms. We will also use the medians of the training columns to fill on the test set as well.

```
## # A tibble: 1 x 15
##   price room_type neighbourhood_group_cleansed instant_bookable bedrooms
##   <int>     <int>                         <int>            <int>    <int>
## 1     0         0                             0                0        0
## # i 10 more variables: minimum_nights <int>, maximum_nights <int>,
## #   latitude <int>, longitude <int>, availability_30 <int>, accommodates <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_is_superhost <int>, review_scores_rating <int>
```

```
## # A tibble: 3 x 15
##   price   room_type       neighbourhood_group_cleansed instant_bookable bedrooms
##   <chr>   <chr>           <chr>                        <lgl>                <dbl>
## 1 $95.00  Entire home/apt Brooklyn                     TRUE                     1
## 2 $231.00 Entire home/apt Brooklyn                     FALSE                    3
## 3 $140.00 Private room    Queens                       FALSE                    1
## # i 10 more variables: minimum_nights <dbl>, maximum_nights <dbl>,
## #   latitude <dbl>, longitude <dbl>, availability_30 <dbl>, accommodates <dbl>,
## #   host_listings_count <dbl>, host_total_listings_count <dbl>,
## #   host_is_superhost <lgl>, review_scores_rating <dbl>
```

Transform price column. remove the dollar sign from the columns

```
## # A tibble: 1 x 15
##   price room_type neighbourhood_group_cleansed instant_bookable bedrooms
##   <int>     <int>                         <int>            <int>    <int>
## 1     0         0                             0                0        0
## # i 10 more variables: minimum_nights <int>, maximum_nights <int>,
## #   latitude <int>, longitude <int>, availability_30 <int>, accommodates <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_is_superhost <int>, review_scores_rating <int>
```

Transform the test set as well . We will use the medians of the training set to fill those NA values on the test set.

```
## # A tibble: 1 x 15
##   price room_type neighbourhood_group_cleansed instant_bookable bedrooms
##   <int>     <int>                        <int>            <int>    <int>
## 1     0         0                            0                0        0
## # i 10 more variables: minimum_nights <int>, maximum_nights <int>,
## #   latitude <int>, longitude <int>, availability_30 <int>, accommodates <int>,
## #   host_listings_count <int>, host_total_listings_count <int>,
## #   host_is_superhost <int>, review_scores_rating <int>
```

```
## [1] 25953
```

```
## [1] 11121
```

Let us investigate the varince inflation factor of each variable. It seems like variables such as neighbourhood_group_cleansed, host_listings_count, host_total_listings_count have VIF of 8 and above. Having them in the model
inflates the variances of the the other variables by more than 8. So it is clear that there is some correlation among these variables.

```
##                                 GVIF Df GVIF^(1/(2*Df))
## room_type                   1.381991  3        1.055401
## neighbourhood_group_cleansed 8.543407  4        1.307537
## instant_bookable            1.074669  1        1.036662
## bedrooms                    1.671451  1        1.292846
## minimum_nights              1.027940  1        1.013874
## maximum_nights              1.000294  1        1.000147
## latitude                    2.809719  1        1.676222
## longitude                   2.946830  1        1.716633
## availability_30             1.123295  1        1.059856
## accommodates                1.941403  1        1.393342
## host_listings_count         8.708636  1        2.951040
## host_total_listings_count   8.775204  1        2.962297
## host_is_superhost           1.064272  1        1.031635
## review_scores_rating        1.027482  1        1.013648
```

Now let us look at the summary to see if there are extreme values. We can see maximum_nights have extreme values that dont make sense. The maximum maximum_nights is 2.147e+09 days. We would normally expect maximum nights to be a year or maybe couple of of years depending on the policy of Airbnb. But this values seems extreme. Although we are not going to remove it from our model (we would expect our machine lerning algorithm to give less emphasis on it during training as there are few observations with that extreme value), it is important to note it here. If we remove it from the training set we also have to remove it from the test set. if we transform it on the training set, we might have to transform it on the test set as well.

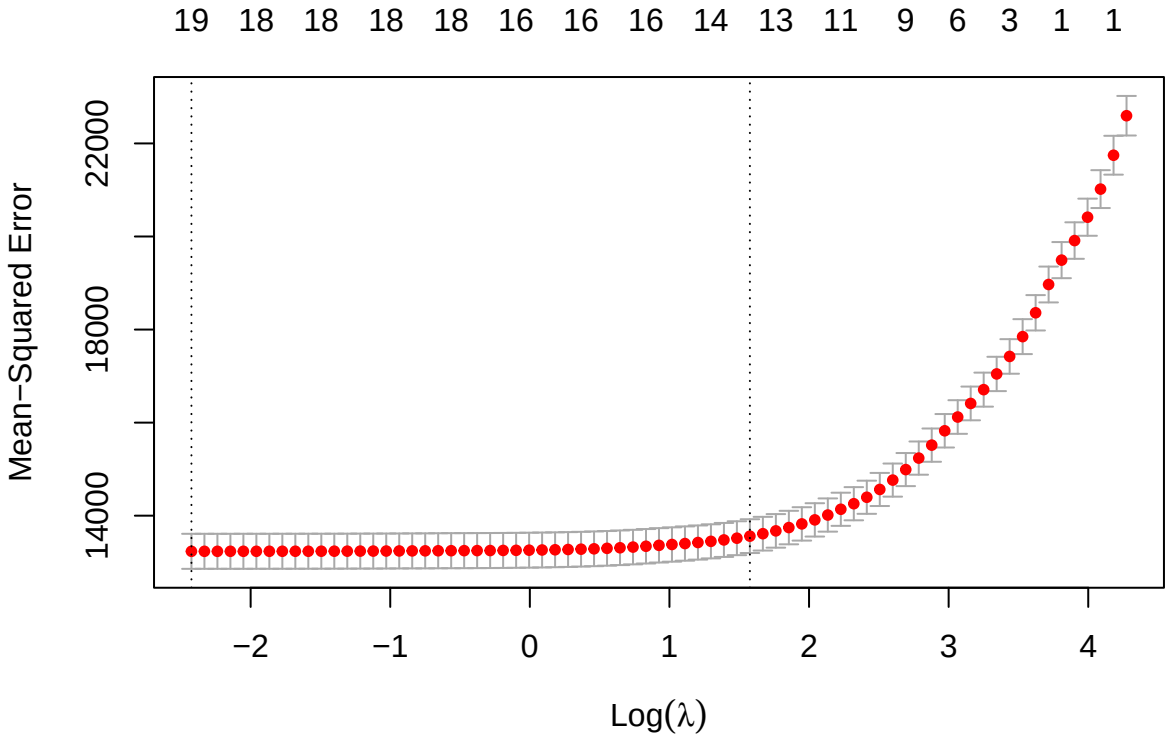
```
##      price        room_type         neighbourhood_group_cleansed
##  Min.   :  8.0   Length:25953       Length:25953
##  1st Qu.: 77.0   Class :character   Class :character
##  Median :130.0   Mode  :character   Mode  :character
##  Mean   :176.5
##  3rd Qu.:220.0
##  Max.   :999.0
##  instant_bookable    bedrooms      minimum_nights    maximum_nights
##  Mode :logical    Min.   : 1.000   Min.   :   1.00   Min.   :1.000e+00
```

```
##  FALSE:21033      1st Qu.: 1.000   1st Qu.:  30.00   1st Qu.:1.800e+02
##  TRUE :4920       Median : 1.000   Median :  30.00   Median :3.650e+02
##                   Mean   : 1.316   Mean   :  30.73   Mean   :8.491e+04
##                   3rd Qu.: 1.000   3rd Qu.:  30.00   3rd Qu.:1.125e+03
##                   Max.   :50.000   Max.   :1250.00   Max.   :2.147e+09
##     latitude       longitude      availability_30   accommodates
##  Min.   :40.50   Min.   :-74.25   Min.   : 0.000   Min.   : 1.000
##  1st Qu.:40.69   1st Qu.:-73.98   1st Qu.: 0.000   1st Qu.: 2.000
##  Median :40.73   Median :-73.95   Median : 0.000   Median : 2.000
##  Mean   :40.73   Mean   :-73.95   Mean   : 7.321   Mean   : 2.822
##  3rd Qu.:40.76   3rd Qu.:-73.93   3rd Qu.:13.000   3rd Qu.: 4.000
##  Max.   :40.91   Max.   :-73.71   Max.   :30.000   Max.   :16.000
##  host_listings_count host_total_listings_count host_is_superhost
##  Min.   :   1.0      Min.   :   1.0            Mode :logical
##  1st Qu.:   1.0      1st Qu.:   1.0            FALSE:20483
##  Median :   2.0      Median :   3.0            TRUE :5470
##  Mean   : 138.1      Mean   : 215.7
##  3rd Qu.:   7.0      3rd Qu.:  10.0
##  Max.   :4559.0      Max.   :8820.0
##  review_scores_rating
##  Min.   :0.000
##  1st Qu.:4.700
##  Median :4.830
##  Mean   :4.675
##  3rd Qu.:4.940
##  Max.   :5.000
```

4 3.Training our machine learning algorithms.

Now we have cleaned and transformed the data. let us go through the training phase using three or four Machine learning algorithms. We will use lasso regression, ridge regression, PCA and random forest regressor.

Let us Find the best lambda for lasso. We will do cross validation on the training set and choose the best lambda parameter. (glmnet by default will do 10 fold cross validation)



The Best lassoModel with the minimum squared error was lambda of 0.089 at MSE of 13234. And the lambda that gives the most regularized model where the cross validated error is with in one standard error of lamdda min is lambda 4.837. It resulted in an cross validated MSE error of 13560. The glmnet package often chooses it as it is more regularized (avoid overfitting more).

```
##
## Call:  cv.glmnet(x = x, y = y, alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  0.089    73   13234 376.4      19
## 1se  4.837    30   13560 363.0      14
```

Let us also see which variables have been chosen by each lasso model. 15 variables (this includes dummy variables that are introduced as part of transformation of catagorical variables such as neighbourhood_group_cleansed) were chosen by lambda 1se.

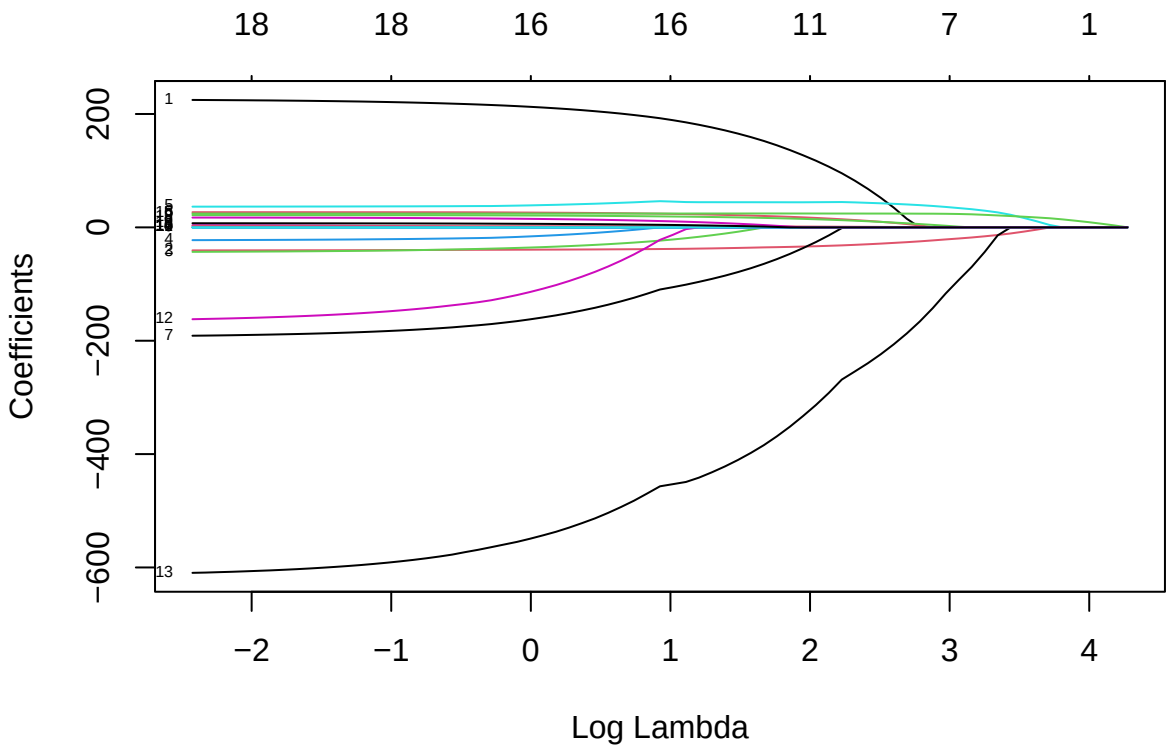
20 variables were chosen by Lambda min.

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                                              s1
## (Intercept)                            -29361.00
## room_typeHotel room                       159.24
## room_typePrivate room                     -35.82
## room_typeShared room                       -4.08
## neighbourhood_group_cleansedBrooklyn        .
## neighbourhood_group_cleansedManhattan      44.33
## neighbourhood_group_cleansedQueens          .
```

```
## neighbourhood_group_cleansedStaten Island    -72.02
## instant_bookableTRUE                          20.77
## bedrooms                                      17.15
## minimum_nights                                -0.28
## maximum_nights                                 .
## latitude                                       .
## longitude                                   -397.89
## availability_30                                1.66
## accommodates                                  24.48
## host_listings_count                            .
## host_total_listings_count                      0.02
## host_is_superhostTRUE                          5.30
## review_scores_rating                           1.87
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                                                 s1
## (Intercept)                                -38427.58
## room_typeHotel room                           224.80
## room_typePrivate room                         -40.62
## room_typeShared room                          -43.18
## neighbourhood_group_cleansedBrooklyn          -22.54
## neighbourhood_group_cleansedManhattan          36.55
## neighbourhood_group_cleansedQueens              3.79
## neighbourhood_group_cleansedStaten Island    -191.16
## instant_bookableTRUE                           26.77
## bedrooms                                       21.66
## minimum_nights                                 -0.41
## maximum_nights                                  0.00
## latitude                                     -162.03
## longitude                                    -609.47
## availability_30                                 2.18
## accommodates                                   24.28
## host_listings_count                            -0.01
## host_total_listings_count                       0.03
## host_is_superhostTRUE                          17.32
## review_scores_rating                            7.26
```

Looking at the plot how the coeefiecnts decrease and gets eliminated as lambda increases.



So based on cross validation we have chosen the lambda values for our lasso regression. We can retrain on the whole training set (as we often do after choosing our best parameters using cross validation) using the lamda min or lambda 1se but glmnet package already does that so no need to do that here. The code down below, will gave us almost the same results on the test set. (we are not running it here)

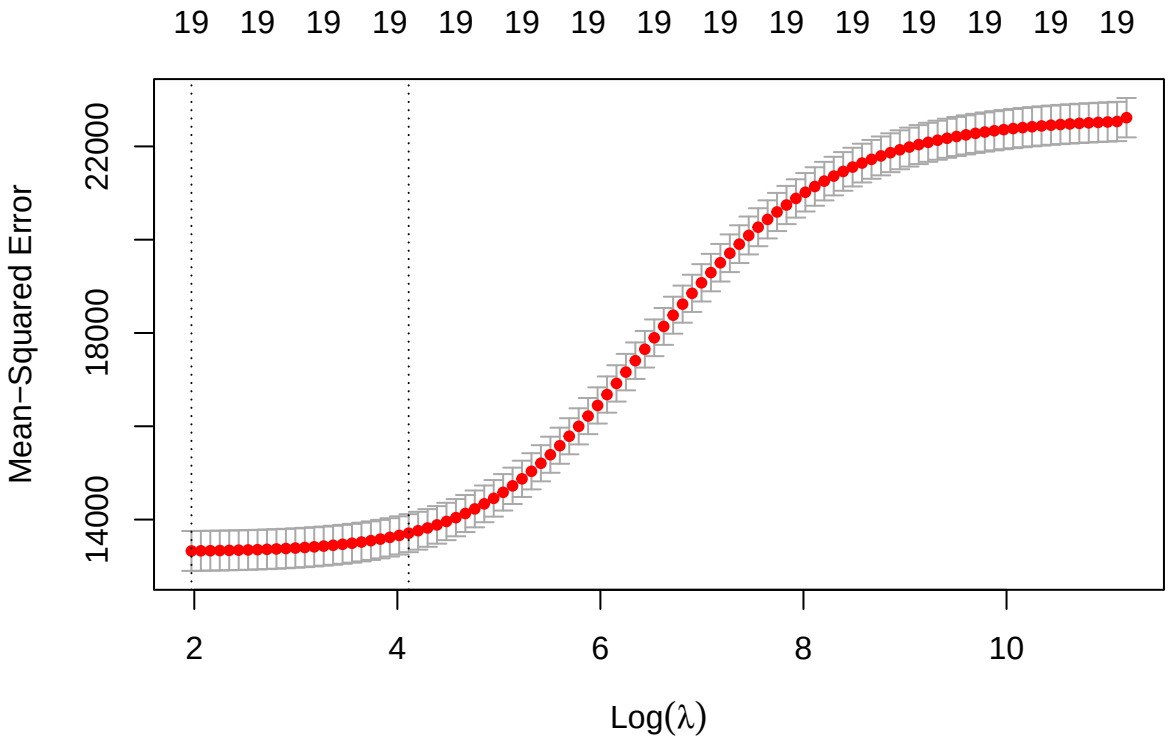
5 Performance of lasso on the test set.

Test it on the test set. We will gauge the performance of lasso using the lambda min and lambda 1se.

lambda.min performed better with MSE error of 13305.35 . So based on its performance on the test set, lambda min of 0.089 is the chosen model for lasso. We will use this performance (13305.35) to check if its performance is better than the best random forest regressor or ridge regression on the test set.

```
## lambda.1se lambda.min
##   13696.48   13305.35
```

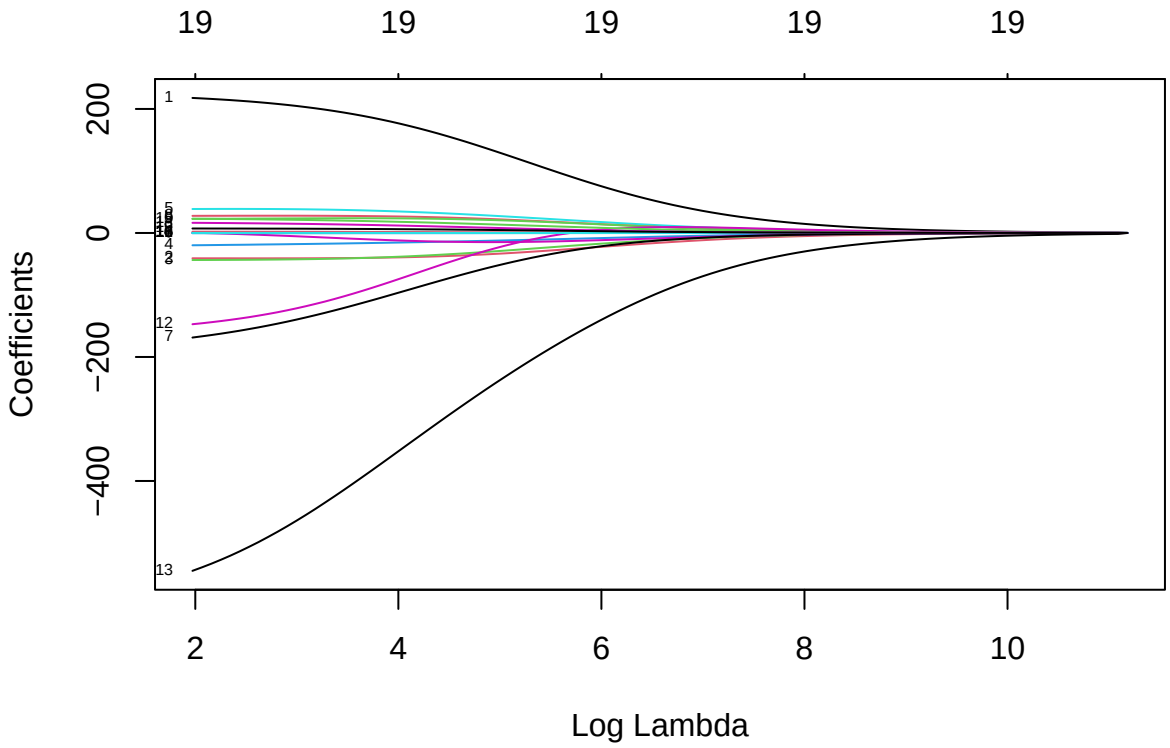
6 Ridge Regression



Ridge Model: The best lamda was 7.18 with a cross validated mean squared error of 13327. The 1se lamda gave us a cross validated error of 13708. All 19 variables (including the dummy ones) were chosen as is the case for a ridge regression.

```
##
## Call:  cv.glmnet(x = x, y = y, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min   7.18   100   13327 427.7      19
## 1se  61.04    77   13708 407.3      19
```

Plot shwing how the coeffiecients decrease as lambda increases.



7 Performance of Ridge on the test set.

Here on the test set (just like the lasoo), the model with lambda min performed better on the test set. The MSE error on the test set for lambda.min was 13323.68 and 13753.10for lambda.1se. So based on its performance on the test set, we will choose the Rddge regression model lambda.min (lambda of 7.18) as our best candidate for ridge with a test set performance of 13323.68 This is slighly higher than what we got for lasso (13305.35). So, so far the lasso model with its minimum lambda is our preferred model based on its performance on the test set.

```
## lambda.1se lambda.min
##   13753.10   13323.68
```

8 Random Forest

Random Forest . Since it was running very slow we only decided to tune the number of trees. We decided to use the default mtry here. The default is square root of total number of features. (square root of 14 which is close to 3). For now using 100 trees (but will increase it to 200 or the default 500 trees). We dont need to do a separate cross validation for random forest regressor as it uses out of bag samples to calculate the out of bag error. There were will be good number of observations that wont be picked in the bootstrapped sample of a tree. So it can treat those out of bag samples as a test set and use the trees that an observation isnt picked in to predict the price for the observation
The key point here is testing on out of bag sample here serves the same purpose that a cross validation will do on validation sets and we get that cheapily by the very nature of bootstrapping in random forest.

Based on that we got a random forest with 99 trees to perform the best.

```
## [1] 100
```

9 Performance of Random Forest on the test set.

.Now let us see its performance on the test set using 100 trees. We got MSE error of 8216.3. So far this is the lowest MSE on the test set compared to lasso regression and Ridge regression. The random forest regressor might have worked better because there might be complex non linear relationships between price and the dependent variables.

```
## [1] 8216.3
```

10 Principal components .

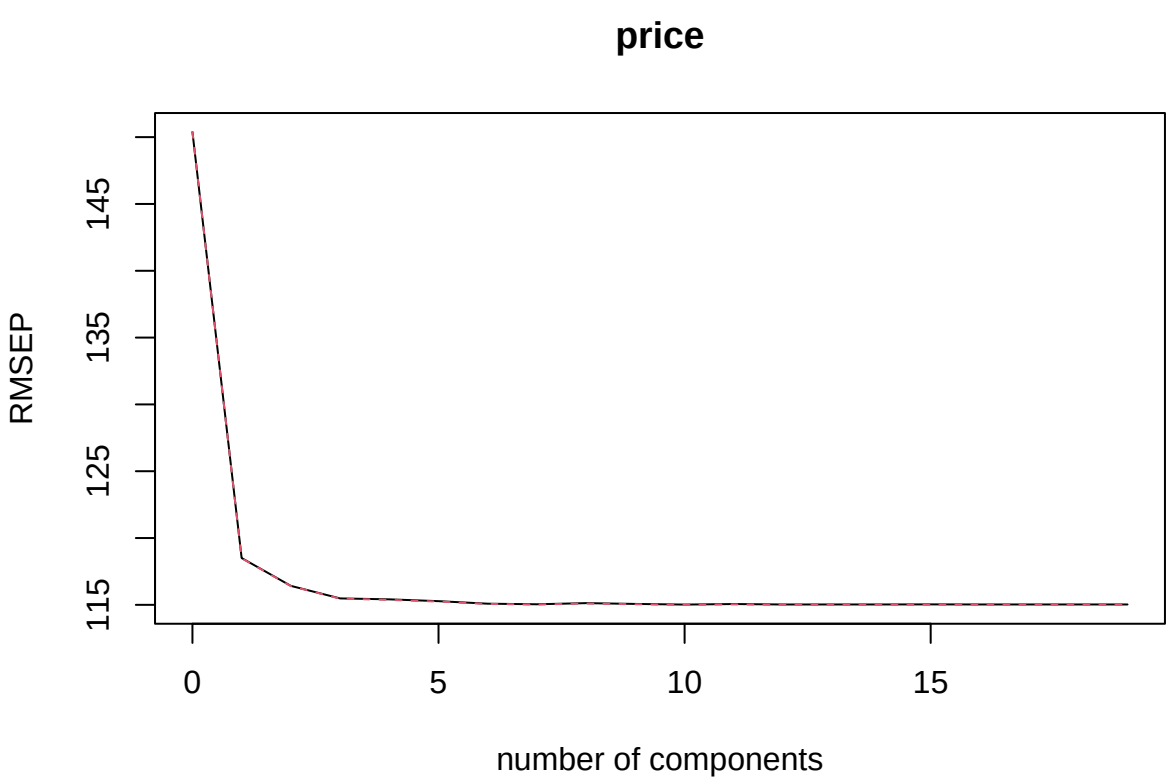
11 To Do

Dhruv part down below:

By running principal component analysis, Specifically partial least square regression, we were able to do cross validation to choose the best number of components. Based on the cross validated error amount or based on the root mean square error on the cross validation, the smallest adjCV was 115. That happened when we use 7 components. So based on cross validation we choose using 7 components for PLSR.

```
## Data:    X dimension: 25953 19
##  Y dimension: 25953 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           150.4    118.5    116.4    115.5    115.4    115.3    115.1
## adjCV        150.4    118.5    116.4    115.5    115.4    115.2    115.1
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV         115    115.1    115.1       115     115.1       115       115
## adjCV      115    115.1    115.0       115     115.0       115       115
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV          115       115       115       115       115       115
## adjCV       115       115       115       115       115       115
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X        11.93    22.70    28.32    33.22    40.93    46.04    49.23    54.38
## price    38.25    40.35    41.34    41.65    41.71    41.76    41.80    41.82
##        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X        59.38     63.93     67.45     70.73     74.07     77.61     79.66
## price    41.83     41.84     41.84     41.84     41.84     41.84     41.84
##        16 comps  17 comps  18 comps  19 comps
## X         84.48     89.29     94.70    100.00
## price     41.84     41.84     41.84     41.84
```

Plotting the vaidation plot graph. looks like the minimum RMSEP (root mean square is around 7 compoennts)



Running it on the test set, we got a mean squared error of 13318.93. This is the highest mean squared error on the test set so far. So we will not be chooosing it as our final model. So far the random forest regressor had the best mean squared error.

```
## [1] 13318.93
```

11.1 conclusion.

Based on the performace on the test set, Random forest with 100 trees was the best performer. For that reason we choose that as our final model.

To recap, we used lasso, ridge, plsr and random forest regressor. For each model, we did cross validation to choose the best lambda for lasso, the best lamda for ridge, the best number of trees for random forest regressor and the best number of components for plsr using the training set and doing 10 fold cross validation. After we choose the best hyparams for these four models, we tested them again on the test set. Then based on the performance on the test set, we choose random forest regressor since it had the minimum mean squared error on the test set.