AD 699 Final Project

Team Boston Cream

2023-12-03

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.0      ✓ purrr   1.0.2
## ✓ tibble  3.2.1      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.5.0
## ✓ readr   2.1.3      ✓ forcats 0.5.2
```

```
## Warning: package 'tibble' was built under R version 4.2.3
```

```
## Warning: package 'purrr' was built under R version 4.2.3
```

```
## ── Conflicts ───────────────────────────────────────── tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
```

```
library(readr)
library(lubridate)
```

```
## Loading required package: timechange
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(leaflet)
```

```
## Warning: package 'leaflet' was built under R version 4.2.3
```

```
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 4.2.3
```

```
## Loading required package: RColorBrewer
```

```
library(textdata)
```

```
## Warning: package 'textdata' was built under R version 4.2.3
```

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.2.3
```

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.2.3
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.2.3
```

```
r <- read_csv('rome_listings.csv')
```

```
## Rows: 28014 Columns: 75
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (25): listing_url, source, name, description, neighborhood_overview, pi...
## dbl  (37): id, scrape_id, host_id, host_listings_count, host_total_listings_...
## lgl   (8): host_is_superhost, host_has_profile_pic, host_identity_verified, ...
## date  (5): last_scraped, host_since, calendar_last_scraped, first_review, la...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
mv <- r %>%
  filter(neighbourhood_cleansed == "XII Monte Verde")
```

```
colSums(is.na(mv))
```

```
##                                         id
##                                          0
##                                 listing_url
##                                          0
##                                   scrape_id
##                                          0
##                                last_scraped
##                                          0
##                                      source
##                                          0
##                                        name
##                                          0
##                                 description
##                                        29
##                       neighborhood_overview
##                                         525
##                                 picture_url
##                                          0
##                                     host_id
##                                          0
##                                    host_url
##                                          0
##                                   host_name
##                                          0
##                                  host_since
##                                          0
##                               host_location
##                                        272
##                                  host_about
##                                        664
##                          host_response_time
##                                          0
##                          host_response_rate
##                                          0
##                       host_acceptance_rate
##                                          0
##                          host_is_superhost
##                                         19
##                          host_thumbnail_url
##                                          0
##                            host_picture_url
##                                          0
##                          host_neighbourhood
##                                        764
##                        host_listings_count
##                                          0
##                     host_total_listings_count
##                                          0
##                             host_verifications
##                                          0
##                           host_has_profile_pic
##                                          0
```

```
##                      host_identity_verified
##                                           0
##                                neighbourhood
##                                         525
##                      neighbourhood_cleansed
##                                           0
##                  neighbourhood_group_cleansed
##                                        1419
##                                    latitude
##                                           0
##                                   longitude
##                                           0
##                                property_type
##                                           0
##                                   room_type
##                                           0
##                                accommodates
##                                           0
##                                   bathrooms
##                                        1419
##                               bathrooms_text
##                                           1
##                                    bedrooms
##                                         319
##                                        beds
##                                          16
##                                   amenities
##                                           0
##                                       price
##                                           0
##                               minimum_nights
##                                           0
##                               maximum_nights
##                                           0
##                      minimum_minimum_nights
##                                           0
##                      maximum_minimum_nights
##                                           0
##                      minimum_maximum_nights
##                                           0
##                      maximum_maximum_nights
##                                           0
##                      minimum_nights_avg_ntm
##                                           0
##                      maximum_nights_avg_ntm
##                                           0
##                            calendar_updated
##                                        1419
##                            has_availability
##                                           0
##                             availability_30
##                                           0
```

```
##                              availability_60
##                                           0
##                              availability_90
##                                           0
##                             availability_365
##                                           0
##                         calendar_last_scraped
##                                           0
##                             number_of_reviews
##                                           0
##                          number_of_reviews_ltm
##                                           0
##                        number_of_reviews_l30d
##                                           0
##                                 first_review
##                                         201
##                                  last_review
##                                         201
##                           review_scores_rating
##                                         201
##                        review_scores_accuracy
##                                         206
##                      review_scores_cleanliness
##                                         206
##                           review_scores_checkin
##                                         206
##                     review_scores_communication
##                                         206
##                          review_scores_location
##                                         206
##                             review_scores_value
##                                         206
##                                     license
##                                        1161
##                               instant_bookable
##                                           0
##                 calculated_host_listings_count
##                                           0
##  calculated_host_listings_count_entire_homes
##                                           0
## calculated_host_listings_count_private_rooms
##                                           0
##  calculated_host_listings_count_shared_rooms
##                                           0
##                              reviews_per_month
##                                         201
```

```
mv1 <- select(mv, id, description, neighborhood_overview, host_id, host_name, host_since, host_r
esponse_time,
              host_response_rate, host_acceptance_rate, host_total_listings_count, host_has_prof
ile_pic,
              host_identity_verified, property_type, accommodates, amenities, price,
              minimum_nights, maximum_nights, has_availability, number_of_reviews,
              review_scores_rating, instant_bookable)

missing_values <- colSums(is.na(mv1))
mvd <- data.frame(Variable = names(missing_values),
                  Missing_Count = missing_values) %>%
  filter(Missing_Count > 0)
mvd
```

```
##                                  Variable Missing_Count
## description                   description            29
## neighborhood_overview neighborhood_overview           525
## review_scores_rating   review_scores_rating           201
```

```
summary(mv1$review_scores_rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00    4.65    4.83    4.71    4.95    5.00     201
```

```
m <-median(mv1$review_scores_rating, na.rm = TRUE)

mv1$description[is.na(mv1$description)] <- ""
mv1$neighborhood_overview[is.na(mv1$neighborhood_overview)] <- ""
mv1$review_scores_rating[is.na(mv1$review_scores_rating)] <- m

colSums(is.na(mv1))
```

```
##                        id               description     neighborhood_overview
##                         0                         0                         0
##                   host_id                 host_name                host_since
##                         0                         0                         0
##        host_response_time       host_response_rate      host_acceptance_rate
##                         0                         0                         0
## host_total_listings_count      host_has_profile_pic    host_identity_verified
##                         0                         0                         0
##             property_type              accommodates                 amenities
##                         0                         0                         0
##                     price            minimum_nights            maximum_nights
##                         0                         0                         0
##          has_availability         number_of_reviews      review_scores_rating
##                         0                         0                         0
##          instant_bookable
##                         0
```

S I I B: We first filter the original data frame so that it only contains our neighborhood and columns that are important for the purpose of this project. Then the output above shows that only description, neighborhood_overview, and review_scores_rating have missing values. For both description and neighborhood_overview, we replace na values with empty space. The reason for doing so is that na in general description usually means there the host write nothing in that section, which is the same as empty space. This is a better approach than just deleting the na rows becausse that means we will lose about 14% of our data which is not ideal. Also, we can still do text mining on these two variables since empty space does not change anything. For review_scores_rating,we first check the summary stats for that column and then decide that it is best to replace missing values with its median value. There do exist 0s for the score which lower its mean and we think median is a more accurate representation of the true average score here.

```
mv1$price <- as.numeric(gsub("[\\$,]", "", mv1$price))

ss <- c(mean(mv1$price), median(mv1$price), min(mv1$price), max(mv1$price), sd(mv1$price))
ss_names <- c('mean', 'median', 'minimum', 'maximum', 'standard deviation')

df <- data.frame(metric = ss_names, value = ss)
df
```

```
##               metric     value
## 1               mean  146.6413
## 2             median  110.0000
## 3            minimum   15.0000
## 4            maximum 9999.0000
## 5 standard deviation  340.7359
```

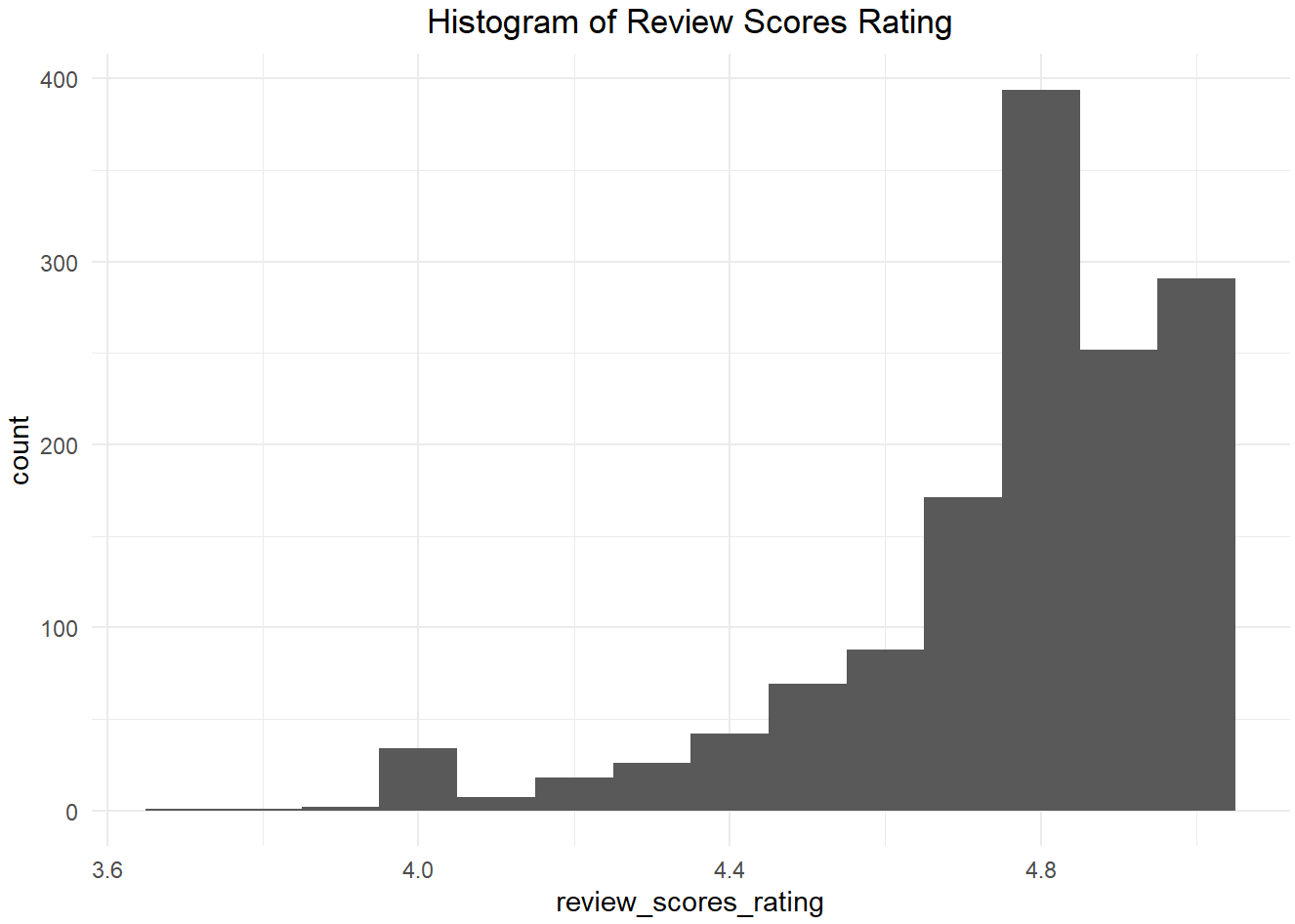
S I II B: The variable we are interested is price. In order to do any analysis on it we first get rid of the dollar sign and convert the column into numeric. The five summary statistics we use here are mean, median, minimum, maximum, and standard deviation. The max price is 9999 which is much higher than the average, even compared to the top 5 highest price. There is a high chance that it is entered incorrectly, maybe it was supposed to be 99.99 instead of 9999. We need to check further on that specific row in order to make final conclusion.

Such high price also has a ribbling effect on the mean and standard deviation, which are 146 and 340. This does not make sense since that will mean any unit that is negative 1 standard deviation away from the mean price will have a negative price, which defies common sense. The median of the price is $110 which is reasonable based on our domain knowledge for this area.

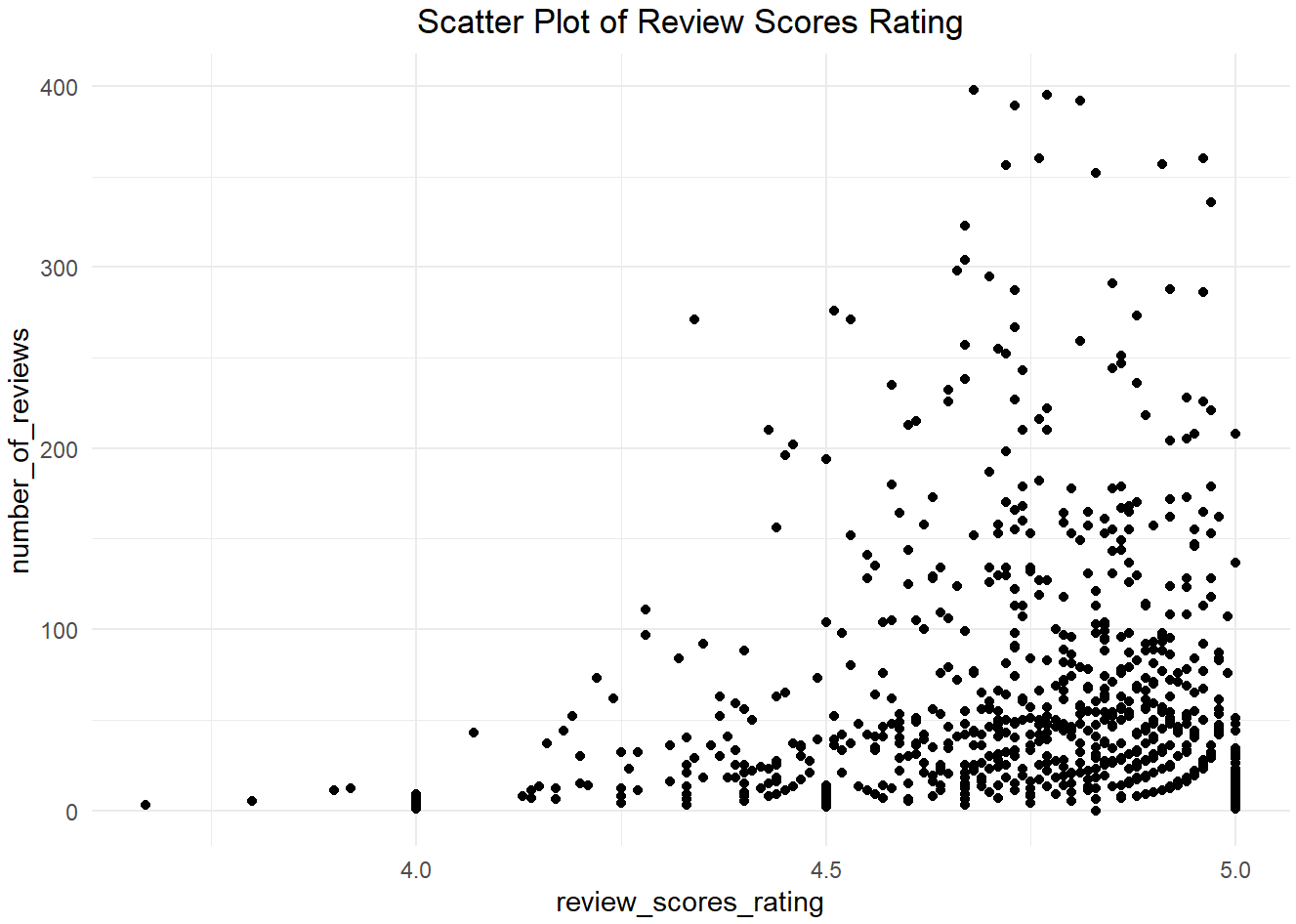
```
# data cleanup and filtering
mv2 <- filter(mv1, review_scores_rating > 3.5)
mv2$instant_bookable <- factor(mv2$instant_bookable)
mv2$host_response_time <- factor(mv2$host_response_time)
mv2$property_type <- factor(mv2$property_type)
mv3 <- mv2 %>%
  mutate(rr = ifelse(host_response_rate == 'N/A', NA , as.numeric(sub("%", "", host_response_rat
e))),
        response_rate_category = cut(rr, breaks = c(0, 20, 40, 60, 80, 100, Inf),
        labels = c("None", "Very Slow", "Slow", "Medium", "Fast", "Very Fast"), include.lowest =
TRUE),
        days_being_host = as.numeric(today() - host_since))
```

```
## Warning in ifelse(host_response_rate == "N/A", NA, as.numeric(sub("%", "", : NAs
## introduced by coercion
```

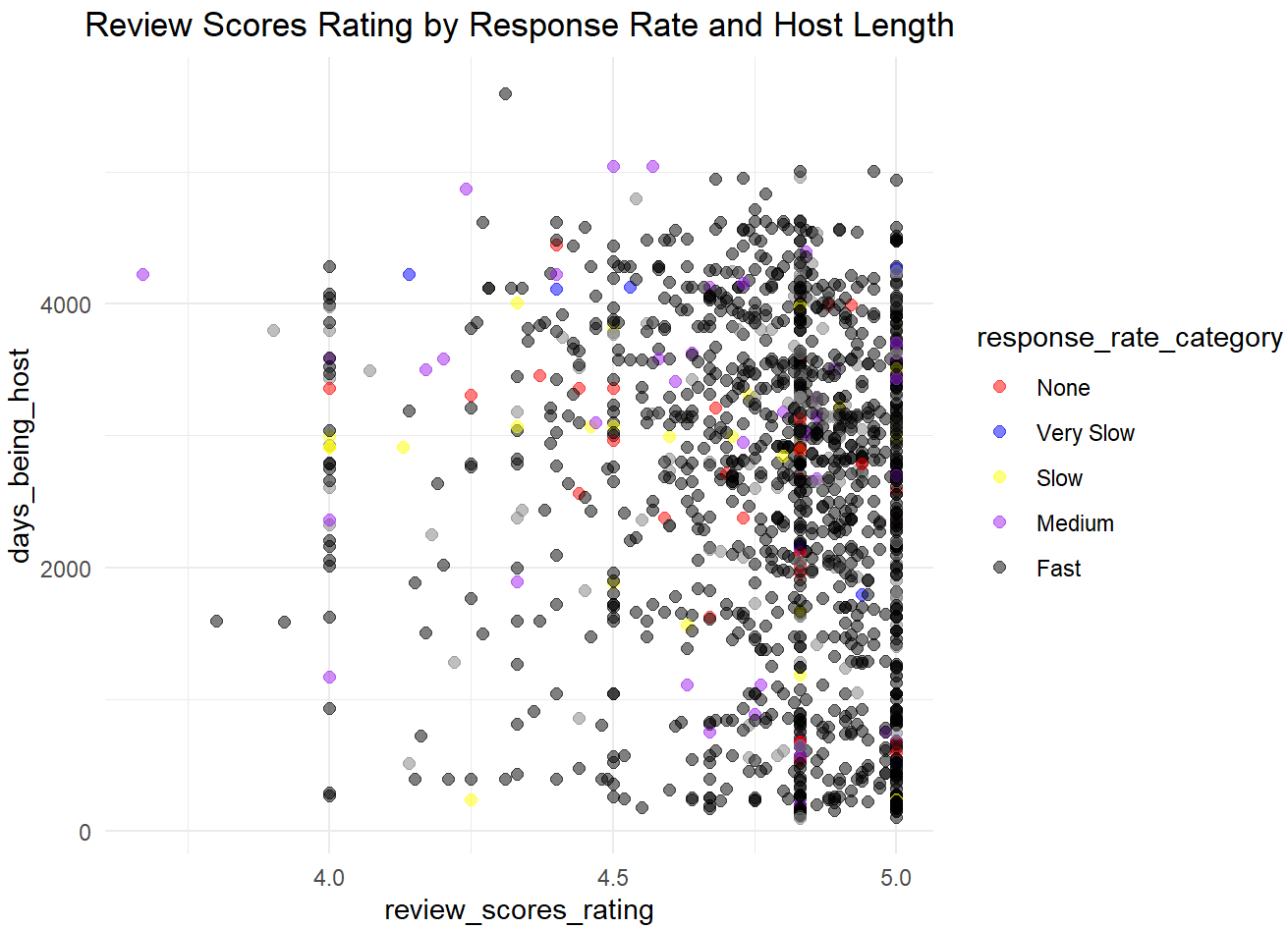
```
# Graphs
ggplot(mv3, aes(x = review_scores_rating)) + geom_histogram(binwidth = 0.1) + theme_minimal() +
  labs(title = 'Histogram of Review Scores Rating') + theme(plot.title = element_text(hjust = 0.
5))
```



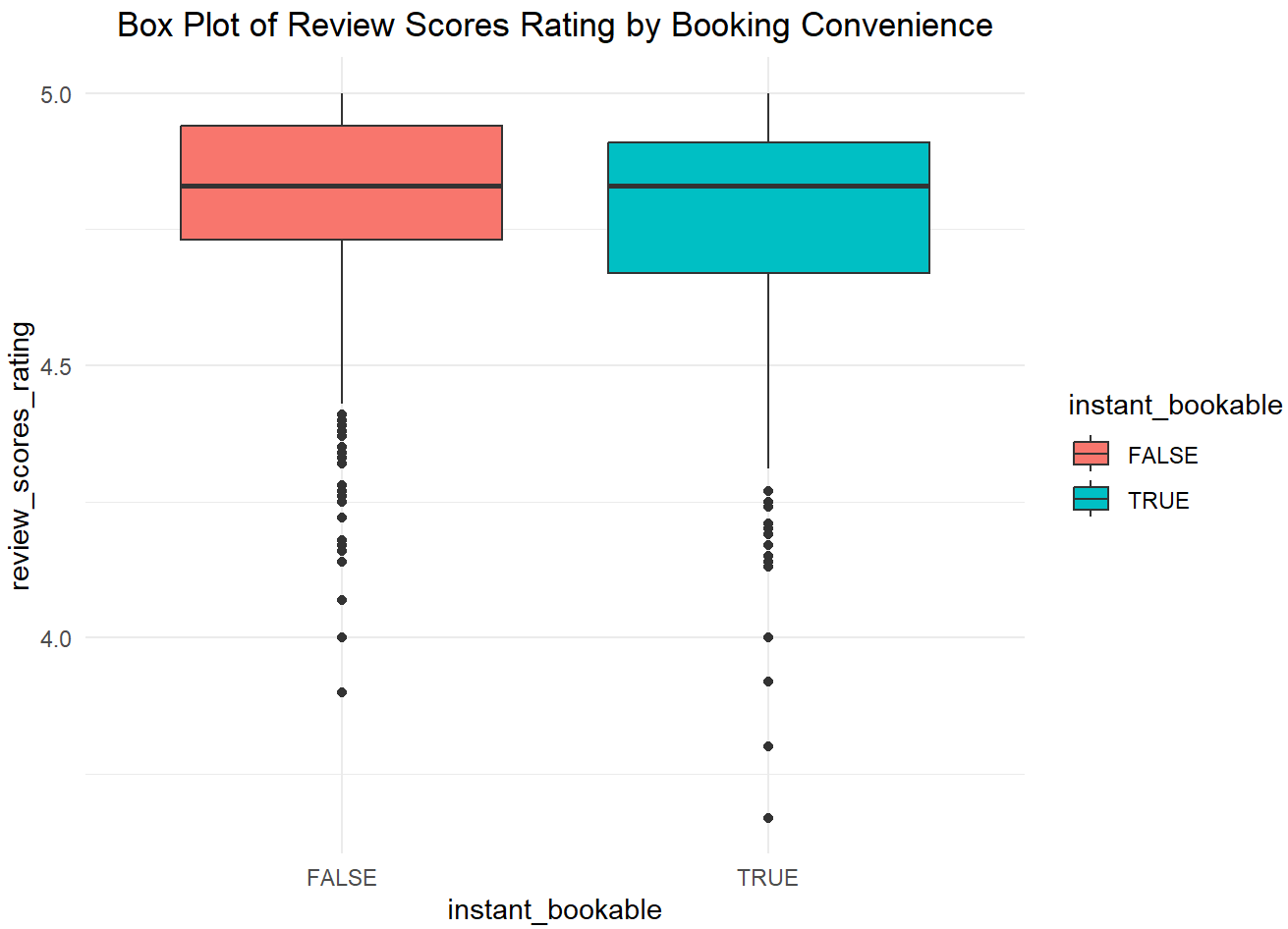
```
ggplot(mv3, aes(x = review_scores_rating, y = number_of_reviews)) + geom_point() + theme_minimal
() +
  labs(title = 'Scatter Plot of Review Scores Rating') + theme(plot.title = element_text(hjust =
0.5))
```



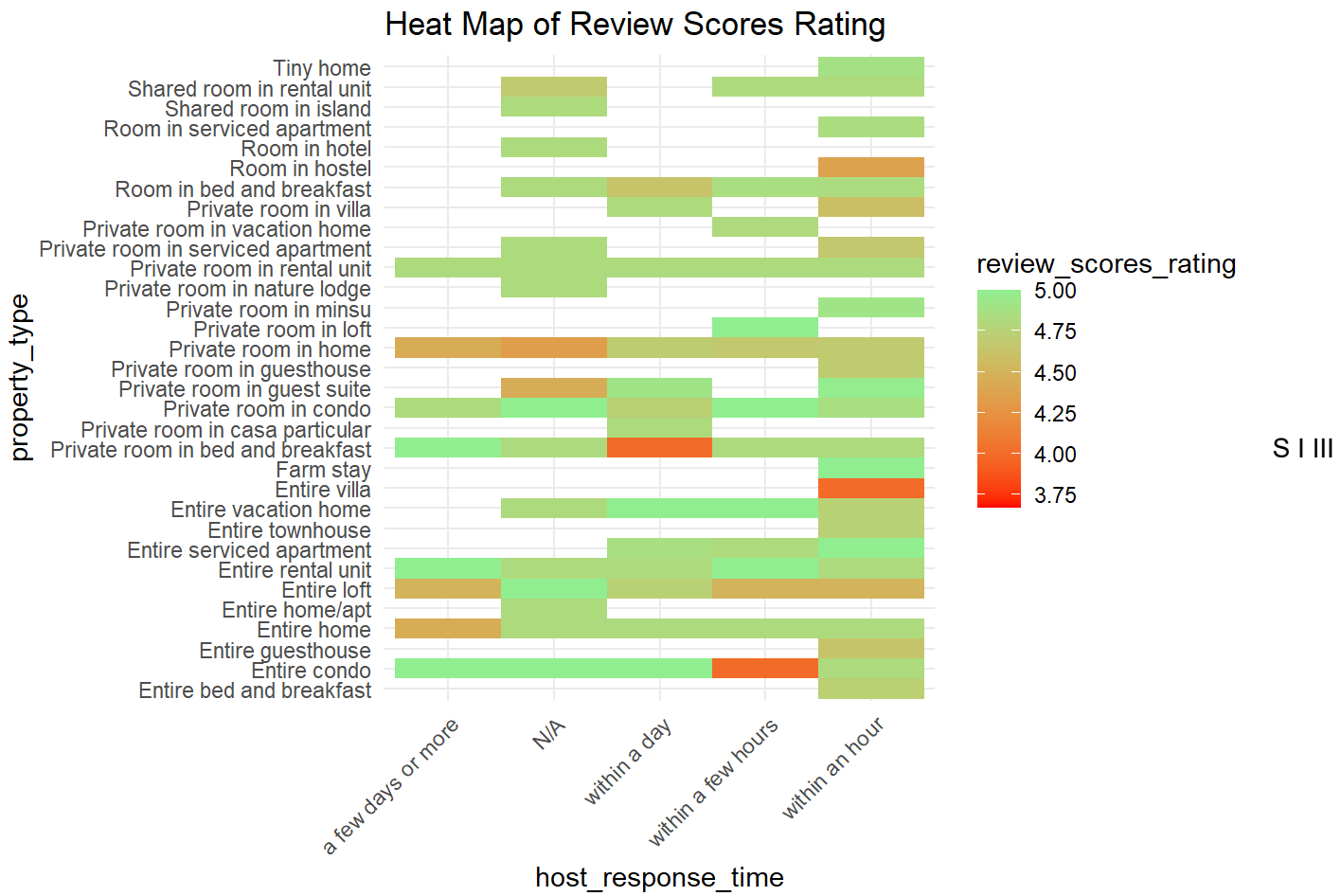

```
cc <- c("None" = 'red', "Very Slow" = 'blue', "Slow" = 'yellow',
        "Medium" = 'purple', "Fast" = 'black', "Very Fast" = 'green')
ggplot(mv3, aes(x = review_scores_rating, y = days_being_host, col = response_rate_category)) +
  geom_point(size = 2, alpha = 0.5) + scale_color_manual(values = cc) + theme_minimal() +
  labs(title = 'Review Scores Rating by Response Rate and Host Length') + theme(plot.title = ele
ment_text(hjust = 0.5))
```




```
ggplot(mv3, aes(x = instant_bookable, y = review_scores_rating, fill = instant_bookable)) + geom
_boxplot() + theme_minimal()+ labs(title = 'Box Plot of Review Scores Rating by Booking Convenie
nce') + theme(plot.title = element_text(hjust = 0.5))
```




```
ggplot(mv3, aes(y = property_type, x = host_response_time, fill = review_scores_rating)) +
  scale_fill_gradient(low = "red", high = "lightgreen") + theme_minimal() + geom_tile() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) + labs(title = 'Heat Map of Review Sc
ores Rating')
```

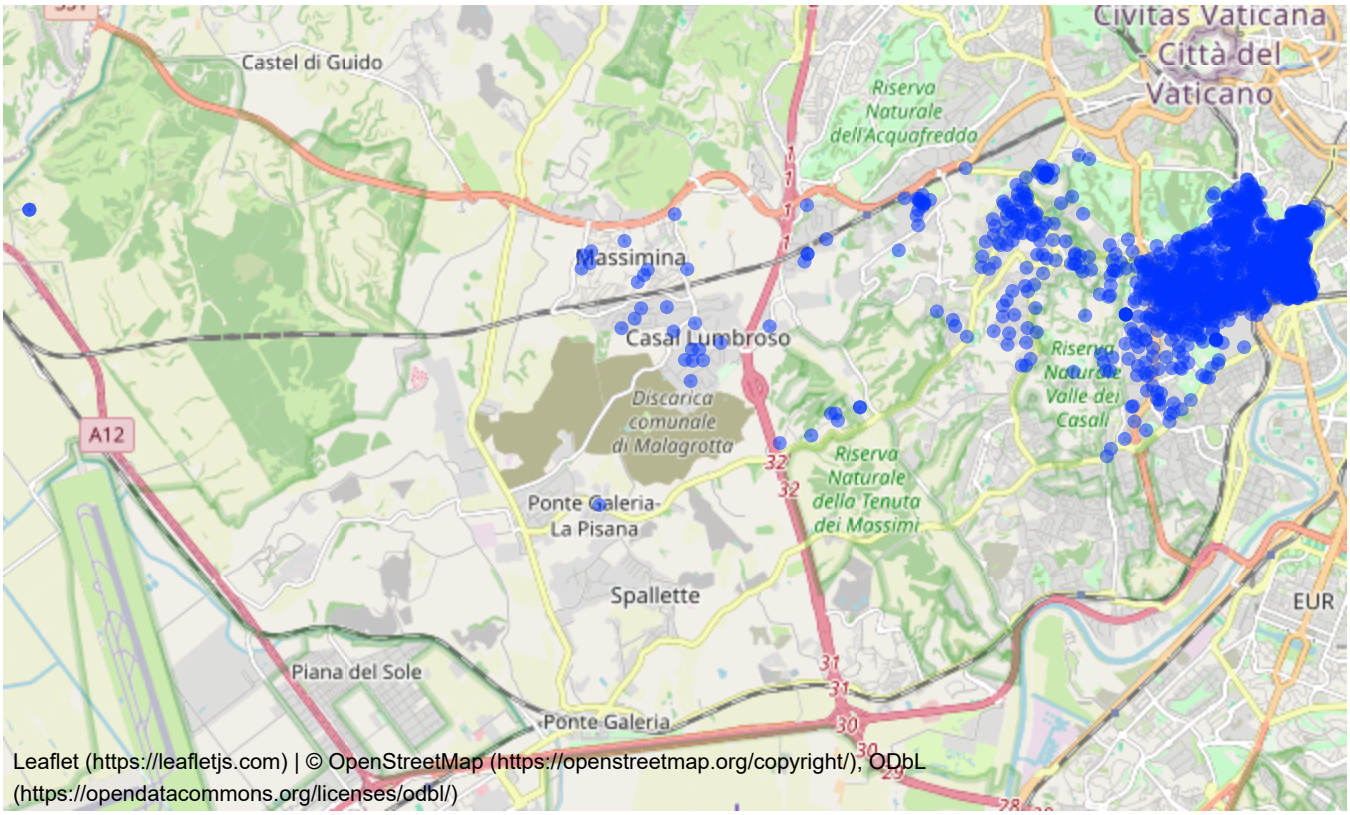


B: The variable we choose here is review_scores_rating. In order to make the graphs more readable, we filter out all the data with score below 3.5(they are outliers anyway and there are not so many of them). The first graph is a histogram shows the distribution of the rating. It seems like majority of the scores are from 4.4 to 5 with peak around 4.8, most of the hosts in our neighborhood have a good reputation. The second graph is a scatter plot of rating and number of reviews. Although the relationship between these two variables do not seem to be very linear, all the units that get 5 do not have that many reviews. Also, most of the units have under 100 reviews.

The third graph is another scatter plot of rating based on response rate and host length. The majority of units that get 5 seem to response very fast. The slow and very slow response tend to get lower ratings, but not by much. Review scores rating also do not seem to be affected by how long the person has been a host. The forth plot is a box plot based on booking convenience. Suprisingly the units that are not instant bookable are getting slightly lower scores. This might be due to the potential lack of communication if a unit could be booked instantly. The last graph is a heat map with respect to response time and property type. We get a consistent result on how response time affect review scores, and we can also see that property type does not influence ratings that much, since there is a relatively high range of scores across all types.

```
m <- leaflet() %>%
  addTiles() %>%
  addCircles(lng= mv$longitude , lat= mv$latitude)

m
```



S I IV: The majority of properties are located in the actual neighborhood, which is what we expected. There is a higher density next to the Tevere river area. This makes sense because people will probably want to enjoy the view. There are also some properties that are located outside of the neighborhood and separated by the Grande Raccordo Anulare. From the map it seems like the landscapes are very different between these two areas.

```
t <- select(mv, neighborhood_overview) %>%
  na.omit() %>%
  unnest_tokens(word, neighborhood_overview)
#  group_by(word) %>%
#  summarize(n = n())

swe <- data.frame(word = stopwords("en"), stringsAsFactors = FALSE) #Italian stop words
swi <- data.frame(word = stopwords("it"), stringsAsFactors = FALSE) #English stop words
extra_sw <- data.frame(word = c('br', 'monteverde'))
sw <- rbind(swe, swi, extra_sw)
w <- anti_join(t, sw, by = "word")

wordcloud(words = w,  min.freq = 150,  scale = c(3.5, 0.5), max.words = 30)
```

S I V:

Some of the key words are ‘trastevere’, ‘quartiere’, ‘rome’, and ‘villa’.

Step 2 Prediction

```
library(dplyr)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

Step 2 Part 1 Task A

```
mv_mlr<-mv
missing_values <- sapply(mv_mlr, function(x) sum(is.na(x)))
missing_values <- sort(missing_values, decreasing = TRUE)
missing_values
```

```
##                 neighbourhood_group_cleansed
##                                         1419
##                                    bathrooms
##                                         1419
##                              calendar_updated
##                                         1419
##                                      license
##                                         1161
##                            host_neighbourhood
##                                          764
##                                   host_about
##                                          664
##                          neighborhood_overview
##                                          525
##                                neighbourhood
##                                          525
##                                     bedrooms
##                                          319
##                                host_location
##                                          272
##                       review_scores_accuracy
##                                          206
##                     review_scores_cleanliness
##                                          206
##                          review_scores_checkin
##                                          206
##                    review_scores_communication
##                                          206
##                         review_scores_location
##                                          206
##                            review_scores_value
##                                          206
##                                 first_review
##                                          201
##                                last_review
##                                          201
##                          review_scores_rating
##                                          201
##                             reviews_per_month
##                                          201
##                                   description
##                                           29
##                             host_is_superhost
##                                           19
##                                        beds
##                                           16
##                               bathrooms_text
##                                            1
##                                           id
##                                            0
##                                   listing_url
##                                            0
```

```
##                                   scrape_id
##                                           0
##                                last_scraped
##                                           0
##                                      source
##                                           0
##                                        name
##                                           0
##                                 picture_url
##                                           0
##                                     host_id
##                                           0
##                                    host_url
##                                           0
##                                   host_name
##                                           0
##                                  host_since
##                                           0
##                           host_response_time
##                                           0
##                           host_response_rate
##                                           0
##                         host_acceptance_rate
##                                           0
##                           host_thumbnail_url
##                                           0
##                             host_picture_url
##                                           0
##                          host_listings_count
##                                           0
##                     host_total_listings_count
##                                           0
##                           host_verifications
##                                           0
##                         host_has_profile_pic
##                                           0
##                       host_identity_verified
##                                           0
##                       neighbourhood_cleansed
##                                           0
##                                     latitude
##                                           0
##                                    longitude
##                                           0
##                                property_type
##                                           0
##                                    room_type
##                                           0
##                                 accommodates
##                                           0
##                                    amenities
##                                           0
```

```
##                                       price
##                                           0
##                                minimum_nights
##                                           0
##                                maximum_nights
##                                           0
##                           minimum_minimum_nights
##                                           0
##                           maximum_minimum_nights
##                                           0
##                           minimum_maximum_nights
##                                           0
##                           maximum_maximum_nights
##                                           0
##                           minimum_nights_avg_ntm
##                                           0
##                           maximum_nights_avg_ntm
##                                           0
##                                 has_availability
##                                           0
##                                  availability_30
##                                           0
##                                  availability_60
##                                           0
##                                  availability_90
##                                           0
##                                 availability_365
##                                           0
##                           calendar_last_scraped
##                                           0
##                               number_of_reviews
##                                           0
##                            number_of_reviews_ltm
##                                           0
##                           number_of_reviews_l30d
##                                           0
##                                 instant_bookable
##                                           0
##                    calculated_host_listings_count
##                                           0
##  calculated_host_listings_count_entire_homes
##                                           0
## calculated_host_listings_count_private_rooms
##                                           0
##  calculated_host_listings_count_shared_rooms
##                                           0
```

```
sum(is.na(mv_mlr))
```

```
## [1] 10592
```

S II I A: In the initial stage of the process, we preprocess the data to prepare it for a regression model, with ‘price’ as the dependent variable. Identifying true NA values throughout the data set is crucial and several variables exhibit NA counts exceeding 500. When we are dealing with these variables, even if the NAs are replaced with the mean, median, or mode, might be unnecessary and introduce uncontrolled effects. In order to mitigate overfitting or underfitting in our model, we opt to drop variables with more than 500 NA values.

Some important variables, such as ‘bedrooms’ or ‘review scores,’ still have NA values. Variables like ‘first review’ or ‘last review’ are unrelated to our pricing model, so we eliminate them in a new data set. Then, we use the median to fill in the NA values for ‘review_scores_rating,’ ‘reviews_per_month,’ ‘bedrooms,’ and ‘beds.’

```
mv_mlr <- mv_mlr %>%
  select(-neighbourhood_group_cleansed, -bathrooms, -calendar_updated,
         -license, -host_neighbourhood, -host_about, -neighborhood_overview,
         -neighbourhood, -host_location, -review_scores_accuracy,
         -review_scores_cleanliness, -review_scores_checkin,
         -review_scores_communication, -review_scores_location,
         -review_scores_value, -first_review, -last_review)

#Replace with median
mv_mlr$review_scores_rating <- ifelse(is.na(mv_mlr$review_scores_rating),
                                      median(mv_mlr$review_scores_rating, na.rm = TRU
E),
                                      mv_mlr$review_scores_rating)

mv_mlr$reviews_per_month <- ifelse(is.na(mv_mlr$reviews_per_month),
                                   median(mv_mlr$reviews_per_month, na.rm = TRUE),
                                   mv_mlr$reviews_per_month)

mv_mlr$bedrooms <- ifelse(is.na(mv_mlr$bedrooms),
                          median(mv_mlr$bedrooms, na.rm = TRUE),
                          mv_mlr$bedrooms)

mv_mlr$beds <- ifelse(is.na(mv_mlr$beds),
                      median(mv_mlr$beds, na.rm = TRUE),
                      mv_mlr$beds)

mv_mlr$price <- gsub("\\$", "", mv_mlr$price)
mv_mlr$price <- gsub(",", "", mv_mlr$price)
mv_mlr$price <- as.numeric(mv_mlr$price)

mv_data <- mv_mlr %>%
  select(price, room_type, accommodates, bedrooms, beds,
         host_is_superhost, review_scores_rating, number_of_reviews,
         availability_365, instant_bookable)
str(mv_data)
```

```
## tibble [1,419 × 10] (S3: tbl_df/tbl/data.frame)
##  $ price               : num [1:1419] 113 120 65 146 160 128 117 30 55 160 ...
##  $ room_type           : chr [1:1419] "Entire home/apt" "Entire home/apt" "Entire home/apt"
"Entire home/apt" ...
##  $ accommodates        : num [1:1419] 7 2 5 3 2 8 4 2 5 2 ...
##  $ bedrooms            : num [1:1419] 2 1 3 1 1 3 3 1 2 1 ...
##  $ beds                : num [1:1419] 6 1 4 4 3 5 3 1 3 1 ...
##  $ host_is_superhost   : logi [1:1419] TRUE FALSE FALSE TRUE FALSE FALSE ...
##  $ review_scores_rating: num [1:1419] 4.73 4.4 4.59 4.77 4.85 4.27 4.73 4.85 4.84 4.85 ...
##  $ number_of_reviews   : num [1:1419] 166 8 40 395 244 32 155 13 95 155 ...
##  $ availability_365    : num [1:1419] 310 361 133 120 358 317 311 0 66 361 ...
##  $ instant_bookable    : logi [1:1419] TRUE FALSE FALSE FALSE FALSE TRUE ...
```

```
mv_data$host_is_superhost <- ifelse(mv_data$host_is_superhost == "TRUE", 1, 0)
mv_data$instant_bookable <- as.numeric(mv_data$instant_bookable)



dummy_room_type <- model.matrix(~ room_type - 1, data = mv_data)
dummy_room_type <- dummy_room_type[, -4]
mv_data <- cbind(mv_data, dummy_room_type)
mv_data <- mv_data %>% dplyr::select(-room_type)

mv_data$log_price <- log(mv_data$price)
variables_to_scale <- c("number_of_reviews", "availability_365")
mv_data[variables_to_scale] <- scale(mv_data[variables_to_scale])
```

S II I A: Additionally, we notice irrelevant variables like scraping time and URLs in a significant part of the data set. These variables are excluded from our linear regression analysis. We select the following variables for our model: ‘price,’ ‘room_type,’ ‘accommodates,’ ‘bedrooms,’ ‘beds,’ ‘host_is_superhost,’ ‘review_scores_rating,’ ‘number_of_reviews,’ ‘availability_365,’ and ‘instant_bookable.’ Although ‘availability_365’ and ‘instant_bookable’ might intuitively correlate with price, we retain them in the first phase. ‘host_is_superhost’ indicates a host’s popularity and extra services, potentially having an indirect relationship with the price. Furthermore, ‘review_scores_rating’ and ‘number_of_reviews’ reflect the property’s popularity, influencing price dynamics.

Next, we transform the character variable into a numeric format, removing the ‘$’ sign and retaining only the numeric string in the values. A log transformation is then applied to the price variable to mitigate the effects of outliers, especially in larger transaction.

```
str(mv_data)
```

```
## 'data.frame':    1419 obs. of  13 variables:
##  $ price                   : num  113 120 65 146 160 128 117 30 55 160 ...
##  $ accommodates            : num  7 2 5 3 2 8 4 2 5 2 ...
##  $ bedrooms                : num  2 1 3 1 1 3 3 1 2 1 ...
##  $ beds                    : num  6 1 4 4 3 5 3 1 3 1 ...
##  $ host_is_superhost       : num  1 0 0 1 0 0 1 0 0 0 ...
##  $ review_scores_rating    : num  4.73 4.4 4.59 4.77 4.85 4.27 4.73 4.85 4.84 4.85 ...
##  $ number_of_reviews       : num  2.0774 -0.502 0.0204 5.8158 3.3507 ...
##  $ availability_365        : num  0.999 1.406 -0.414 -0.517 1.382 ...
##  $ instant_bookable        : num  1 0 0 0 0 1 1 0 0 0 ...
##  $ room_typeEntire home/apt: num  1 1 1 1 0 1 1 1 1 0 ...
##  $ room_typeHotel room     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ room_typePrivate room   : num  0 0 0 0 1 0 0 0 0 1 ...
##  $ log_price               : num  4.73 4.79 4.17 4.98 5.08 ...
```

```
mv_data<- na.omit(mv_data)
sum(is.na(mv_data))
```

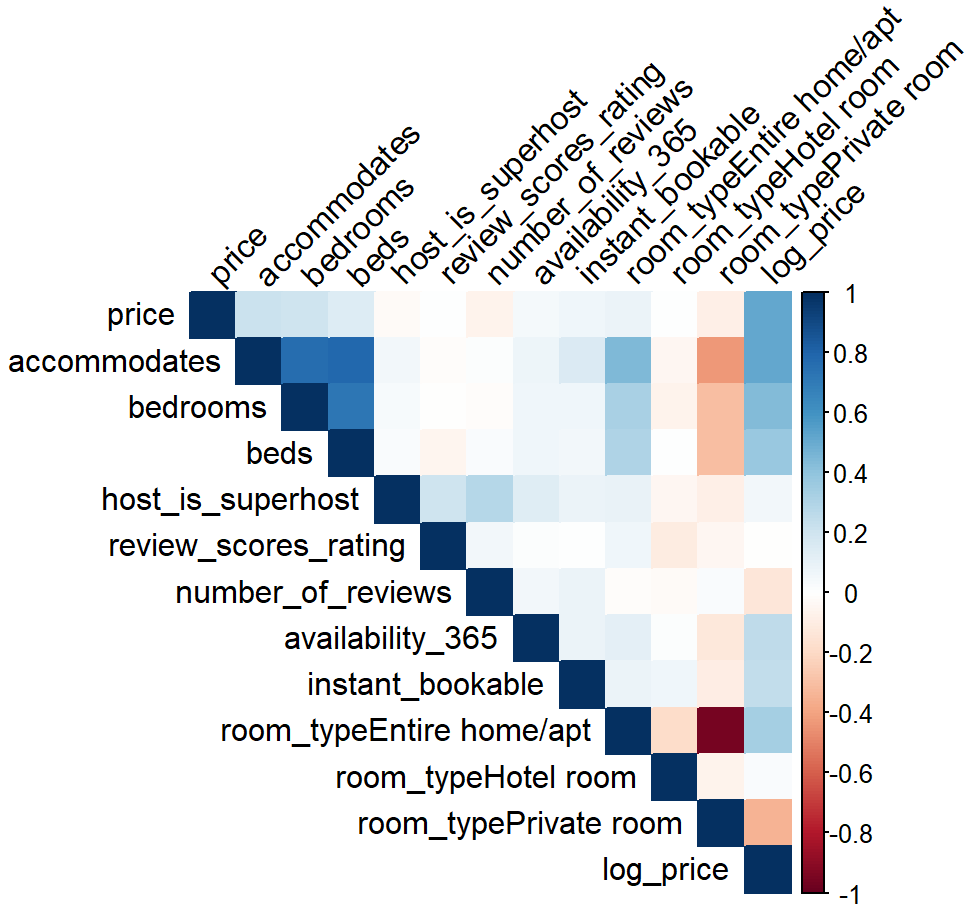
```
## [1] 0
```

Model evaluation involves several metrics such as R-squared, RMSE, and MAE for both training and validation. A reasonable R-squared is prioritized over a simple high R-squared. A 0.05 p-value is used for further tuning to ensure the precision of the model and check the relationship between predicting variables and the dependent variable, price.

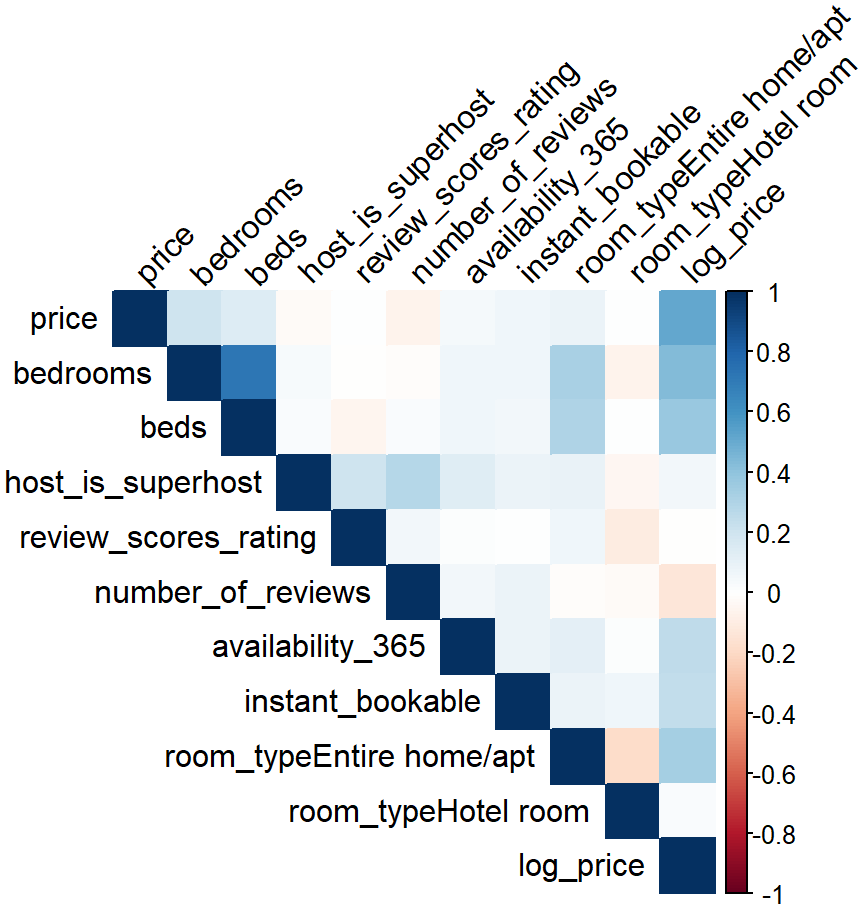
```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
cor_matrix <- cor(mv_data)
# Create a correlation plot
corrplot(cor_matrix, method = "color", type = "upper", tl.col = "black", tl.srt = 45,number.cex
= 0.7)
```




```
mv_data1 <- mv_data[, !colnames(mv_data) %in% c("accommodates", "room_typePrivate room")]
cor_matrix1 <- cor(mv_data1)
# Create a correlation plot
corrplot(cor_matrix1, method = "color", type = "upper", tl.col = "black", tl.srt = 45,number.cex
= 0.7)
```



S II I

A: Before fitting all variables into the model, we check for collinearity among them using a correlation plot. A correlation higher than 0.6 is considered significant. From the plot, we observe high correlations among ‘accommodates’ and ‘beds,’ ‘accommodates’ and ‘bedrooms,’ and ‘room_typePrivate room’ and ‘room_type entire room.’ Consequently, we decide to exclude ‘accommodates’ and ‘room_typePrivate room,’ resulting in no high correlation in our price model.

Step 2 Part 1 Task B

```
set.seed(699)
train.index <- sample(c(1:nrow(mv_data1)), nrow(mv_data1)*0.59)

# Split the data into training and validation sets
train.df <- mv_data1[train.index, ]
valid.df <- mv_data1[-train.index, ]

model <- lm(log_price ~ . - price, data = train.df)
# Summary of the model
summary(model)
```

```
##
## Call:
## lm(formula = log_price ~ . - price, data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76320 -0.30066  0.00095  0.26029  2.91210
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3.77929    0.20153  18.753  < 2e-16 ***
## bedrooms                    0.27113    0.03327   8.149 1.38e-15 ***
## beds                        0.03728    0.01801   2.070   0.0388 *
## host_is_superhost           0.01825    0.03928   0.465   0.6423
## review_scores_rating        0.03091    0.04183   0.739   0.4601
## number_of_reviews          -0.07905    0.01777  -4.449 9.83e-06 ***
## availability_365            0.12548    0.01700   7.382 3.84e-13 ***
## instant_bookable            0.17953    0.03464   5.183 2.76e-07 ***
## `room_typeEntire home/apt`  0.26160    0.04268   6.130 1.37e-09 ***
## `room_typeHotel room`       0.29225    0.17498   1.670   0.0953 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4828 on 816 degrees of freedom
## Multiple R-squared:  0.371,  Adjusted R-squared:  0.3641
## F-statistic: 53.48 on 9 and 816 DF,  p-value: < 2.2e-16
```

S II I B: We have our final MLR model and its summary shown in the table above. The whole equation is describe as: 3.77929(intercept)+0.27113* bedrooms+0.03728* beds+0.01825* host_is_superhost+0.03091* number_of_reviews-0.07905* number_of_reviews+ 0.12548* availability_365+0.17953* instant_bookable+0.26160* room_typeEntire home/apt+ 0.29225* room_typeHotel room

In the model, bedrooms, beds, number_of_reviewsand availability_365 are numerical variables which number_of_reviews and availability_365 are both scaled to make the model less overfitting. The rest are binary values which are transferred from categorical variables.

The regression summary reveals some variables with higher p-values than our threshold of 0.05. ‘host_is_superhost,’ ‘review_scores_rating,’ and ‘room_typeHotel room’ are not statistically significant to the dependent variable ‘price.’ Notably, ‘bedrooms’ has a coefficient of 0.271, aligning with the intuitive understanding that more rooms impact pricing. Surprisingly, ‘number_of_reviews’ is associated with a decrease in price.we have Multiple R-squared: 0.371 is close enough for our estimate since it is a raw data with a lot of missing variables. A high r squared would result in overfitting problem.

```
backward <- step(model, direction='backward', scope=formula(model), trace=0)
summary(backward)
```

```
##
## Call:
## lm(formula = log_price ~ bedrooms + beds + number_of_reviews +
##     availability_365 + instant_bookable + `room_typeEntire home/apt` +
##     `room_typeHotel room`, data = train.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76003 -0.29673  0.00272  0.26068  2.91064
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3.92973    0.04376  89.812  < 2e-16 ***
## bedrooms                    0.27114    0.03325   8.154 1.32e-15 ***
## beds                        0.03656    0.01798   2.033   0.0424 *
## number_of_reviews          -0.07600    0.01706  -4.454 9.61e-06 ***
## availability_365            0.12676    0.01686   7.516 1.48e-13 ***
## instant_bookable            0.18018    0.03451   5.222 2.25e-07 ***
## `room_typeEntire home/apt`  0.26559    0.04235   6.272 5.77e-10 ***
## `room_typeHotel room`       0.28318    0.17457   1.622   0.1052
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4825 on 818 degrees of freedom
## Multiple R-squared:  0.3703, Adjusted R-squared:  0.3649
## F-statistic: 68.71 on 7 and 818 DF,  p-value: < 2.2e-16
```

We perform backward elimination in oder to remove predictors with the highest p-values results in the exclusion of ‘host_is_superhost,’ ‘review_scores_rating,’ and ‘room_typeHotel room.’ This slightly increases R-squared but has minimal impact on adjusted R-squared.

Step 2 Part 1 Task C

```
train_predictions <- exp(predict(backward, newdata=train.df))
valid_predictions <- exp(predict(backward, newdata=valid.df))

# Assess accuracy for training set with original price scale
train_accuracy <- accuracy(train_predictions, train.df$price )
train_accuracy
```

```
##                ME     RMSE      MAE      MPE     MAPE
## Test set 21.73021 251.6526 55.08459 -11.5602 38.57608
```

```
# Assess accuracy for test set with original price scale
valid_accuracy <- accuracy(valid_predictions, valid.df$price)
valid_accuracy
```

```
##                ME    RMSE      MAE       MPE    MAPE
## Test set 43.47765 427.852 71.64218 -4.864977 37.1256
```

S II I C: Further performance analysis indicates a mean error of 21.73 on the training set, suggesting a slight overprediction tendency. The RMSE and MAE values are 251.65 and 55.08, respectively. On the test set, the model shows a mean error of 43.48, higher RMSE (427.85) and MAE (71.64), with a lower MPE (-4.86%) and a similar MAPE (37.13%). These results signal a need for refinement to enhance predictive accuracy, especially on unseen data.

Step 3 Classification Step 3 Part 1 Task A

```
mv$price <- as.numeric(gsub("[\\$,]", "", mv$price))
#extract beds and bedrooms
pattern_bedrooms <- ".*?(\\d+)\\s*bedroom.*"
pattern_beds <- ".*?(\\d+)\\s*beds.*"
mv$number_of_bedrooms <- as.numeric(gsub(pattern_bedrooms, "\\1", mv$name, perl = TRUE))
```

```
## Warning: NAs introduced by coercion
```

```
mv$number_of_beds <- as.numeric(gsub(pattern_beds, "\\1", mv$name, perl = TRUE))
```

```
## Warning: NAs introduced by coercion
```

```
# Replace NA with 0 if needed (assuming no mention means 0)
mv$number_of_bedrooms[is.na(mv$number_of_bedrooms)] <- 0
mv$number_of_beds[is.na(mv$number_of_beds)] <- 0
contains_one_bed <- grepl("1 bed", mv$name)
mv$number_of_beds[contains_one_bed]<-1
mv_bed<-mv%>%select(name,number_of_bedrooms,number_of_beds)
```

```
knn_df <- mv %>% select(c('amenities', 'price', 'accommodates', 'number_of_bedrooms', 'number_of
_beds'
                          , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms'))

colSums(is.na(knn_df))
```

```
##                                     amenities
##                                             0
##                                         price
##                                             0
##                                  accommodates
##                                             0
##                            number_of_bedrooms
##                                             0
##                                number_of_beds
##                                             0
## calculated_host_listings_count_private_rooms
##                                             0
##  calculated_host_listings_count_shared_rooms
##                                             0
```

```
condition <-grepl('kitchen', knn_df$amenities, ignore.case=TRUE)
knn_df$amenities <- as.integer(condition)
knn_df$amenities <- as.factor(knn_df$amenities)

set.seed(699)
train.index <- sample(c(1:nrow(knn_df)), nrow(knn_df)*0.6)
train.df <- knn_df[train.index, ]
valid.df <- knn_df[-train.index, ]

class_0 <- filter(train.df, amenities==0)
class_1 <- filter(train.df, amenities==1)

t.test(class_0$price, class_1$price)
```

```
##
##  Welch Two Sample t-test
##
## data:  class_0$price and class_1$price
## t = -1.9325, df = 715.18, p-value = 0.0537
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -63.3220894   0.5011299
## sample estimates:
## mean of x mean of y
##  122.3689  153.7794
```

```
t.test(class_0$accommodates, class_1$accommodates)
```

```
##
##  Welch Two Sample t-test
##
## data:  class_0$accommodates and class_1$accommodates
## t = -8.2955, df = 168.65, p-value = 3.302e-14
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.4843203 -0.9136601
## sample estimates:
## mean of x mean of y
##  2.524272  3.723262
```

```
t.test(class_0$number_of_bedrooms, class_1$number_of_bedrooms)
```

```
##
##  Welch Two Sample t-test
##
## data:  class_0$number_of_bedrooms and class_1$number_of_bedrooms
## t = -8.1837, df = 283.19, p-value = 9.43e-15
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.4758155 -0.2913046
## sample estimates:
## mean of x mean of y
##   1.07767   1.46123
```

```
t.test(class_0$number_of_beds, class_1$number_of_beds)
```

```
##
##  Welch Two Sample t-test
##
## data:  class_0$number_of_beds and class_1$number_of_beds
## t = -9.9288, df = 398.69, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.9094800 -0.6088472
## sample estimates:
## mean of x mean of y
##  1.116505  1.875668
```

```
t.test(class_0$calculated_host_listings_count_private_rooms, class_1$calculated_host_listings_co
unt_private_rooms)
```

```
##
##  Welch Two Sample t-test
##
## data:  class_0$calculated_host_listings_count_private_rooms and class_1$calculated_host_listi
ngs_count_private_rooms
## t = 10.638, df = 118.17, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1.699556 2.477018
## sample estimates:
## mean of x mean of y
## 2.6310680 0.5427807
```

```
t.test(class_0$calculated_host_listings_count_shared_rooms, class_1$calculated_host_listings_cou
nt_shared_rooms)
```

```
##
##  Welch Two Sample t-test
##
## data:  class_0$calculated_host_listings_count_shared_rooms and class_1$calculated_host_listin
gs_count_shared_rooms
## t = -2.1432, df = 747, p-value = 0.03242
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.0204917578 -0.0008986166
## sample estimates:
##  mean of x  mean of y
## 0.00000000 0.01069519
```

```
library('caret')
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
train.norm.df <- train.df
valid.norm.df <- valid.df
norm.values <- preProcess(train.df[, c('price', 'accommodates', 'number_of_bedrooms', 'number_of
_beds'
                          , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')],
                          method=c('center', 'scale'))
train.norm.df[, c('price', 'accommodates', 'number_of_bedrooms', 'number_of_beds'
                          , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')] <-
  predict(norm.values, train.df[, c('price', 'accommodates', 'number_of_bedrooms', 'number_of_be
ds'
                          , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')])
valid.norm.df[, c('price', 'accommodates', 'number_of_bedrooms', 'number_of_beds'
                          , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')] <-
  predict(norm.values, valid.df[, c('price', 'accommodates', 'number_of_bedrooms', 'number_of_be
ds'
                          , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')])

library('FNN')
```

```
## Warning: package 'FNN' was built under R version 4.2.3
```

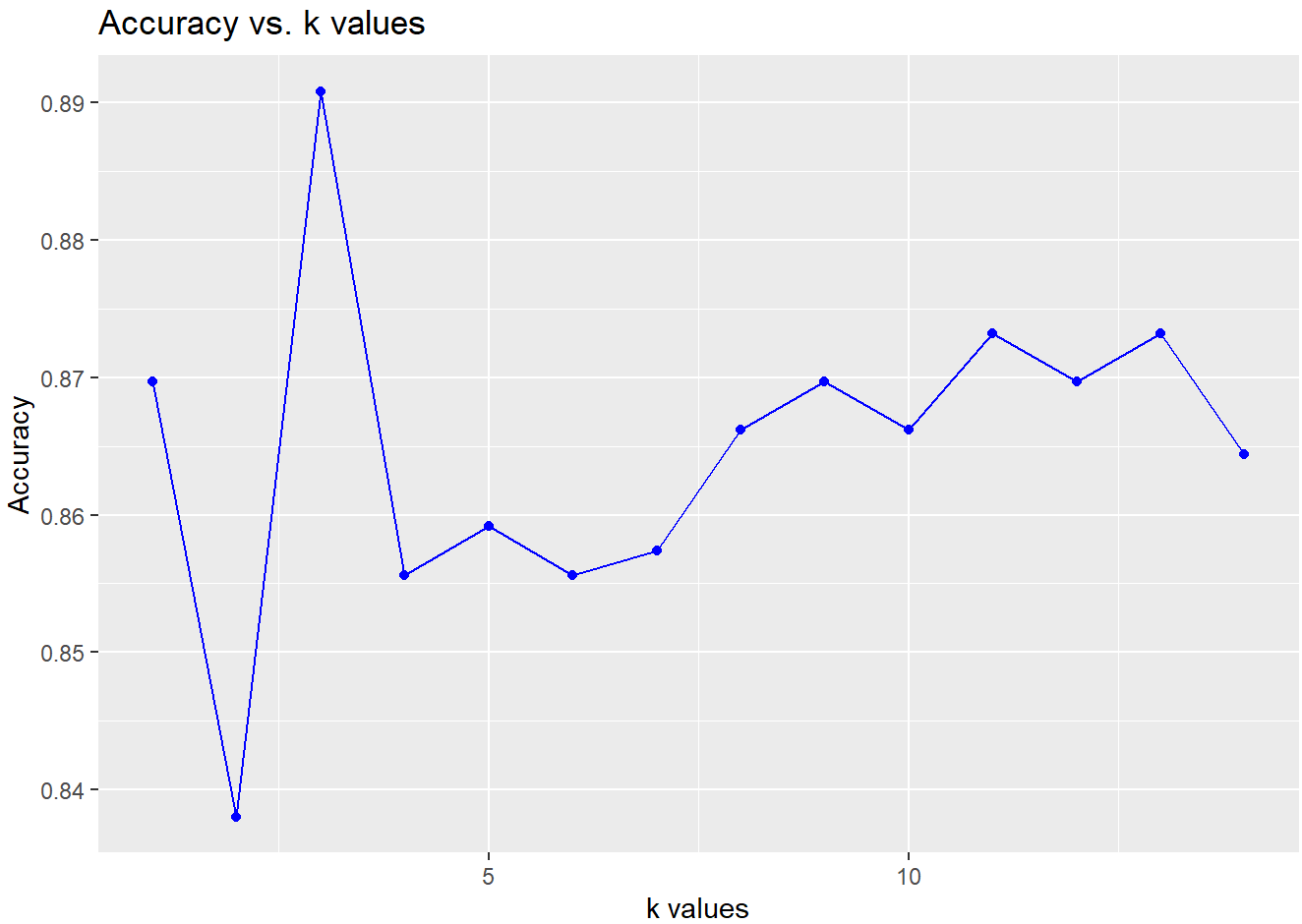
```
accuracy.df <- data.frame(k=seq(1, 14, 1), accuracy = rep(0, 14))

for (i in 1:14) {
  knn.pred <- knn(train.norm.df[,c('price', 'accommodates', 'number_of_bedrooms', 'number_of_bed
s'
                          , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')], valid.norm.df[, c('price', 'accommodates', 'number_of_bedrooms', 'n
umber_of_beds'
                          , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')],
 cl= train.norm.df$amenities , k=i)

  accuracy.df[i,2] <- confusionMatrix(knn.pred, valid.norm.df$amenities)$overall[1]
}
accuracy.df
```

```
##     k  accuracy
## 1   1 0.8697183
## 2   2 0.8380282
## 3   3 0.8908451
## 4   4 0.8556338
## 5   5 0.8591549
## 6   6 0.8556338
## 7   7 0.8573944
## 8   8 0.8661972
## 9   9 0.8697183
## 10 10 0.8661972
## 11 11 0.8732394
## 12 12 0.8697183
## 13 13 0.8732394
## 14 14 0.8644366
```

```
ggplot(accuracy.df, aes(x=k, y=accuracy)) +
  geom_point(color="blue") +
  geom_line(color="blue", aes(group=1)) +
  ggtitle("Accuracy vs. k values") +
  xlab("k values") +
  ylab("Accuracy")
```



```
  knn.pred <- knn(train.norm.df[,c('price', 'accommodates', 'number_of_bedrooms', 'number_of_bed
s'
                           , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')], valid.norm.df[, c('price', 'accommodates', 'number_of_bedrooms', 'n
umber_of_beds'
                           , 'calculated_host_listings_count_private_rooms', 'calculated_host_lis
tings_count_shared_rooms')],
 cl= train.norm.df$amenities , k=3)

confusionMatrix(knn.pred, valid.norm.df$amenities, positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  29  20
##          1  42 477
##
##                Accuracy : 0.8908
##                  95% CI : (0.8623, 0.9153)
##     No Information Rate : 0.875
##     P-Value [Acc > NIR] : 0.139725
##
##                   Kappa : 0.4246
##
##  Mcnemar's Test P-Value : 0.007653
##
##             Sensitivity : 0.9598
##             Specificity : 0.4085
##          Pos Pred Value : 0.9191
##          Neg Pred Value : 0.5918
##              Prevalence : 0.8750
##          Detection Rate : 0.8398
##    Detection Prevalence : 0.9137
##       Balanced Accuracy : 0.6841
##
##        'Positive' Class : 1
##
```

Step III: Classification Part I B: To begin with, we start thinking of the target variable which is amenities. The amenities column contains all unique value which is hard to predict for any model. Thus, we decide to choose kitchen as the outcome class since having a own kitchen is usually the signal of luxury and big apartments.

After dealing with outcome classes, feature selection becomes our priority. Since K nearest neighbor is distance-based model, numeric inputs are the only type that is acceptable under this model. Six features that seems to be logical for predicting luxury apartments were our first choice. After implementing t test of each of those features, all six features are left with contributions to the predictive power. Moreover, we used the above process to find and plot the best k for our model. It turns out that 3 is the best k parameter. According to the confusion matrix of the knn model, the accuracy is 89.1% which is relatively good here. In addition, the TPR is 96% which shows a perfect performance of predicting positive class.

Step 3 Part 2 Task A

```
mv_nb<-mv %>% select(review_scores_rating,host_is_superhost,host_response_time,host_acceptance_r
ate,room_type)
mv_nb<-subset(mv_nb, !is.na(host_is_superhost))
breaks_mam <- quantile(mv_nb$review_scores_rating, probs = seq(0, 1, 1/3), na.rm = TRUE)
mv_nb$review_scores_rating<-cut(mv_nb$review_scores_rating,breaks=breaks_mam,
                                labels = c("low","mid","high"),
                                include.lowest = TRUE)
mv_nb$review_scores_rating <- factor(mv_nb$review_scores_rating, levels = c(levels(mv_nb$review_
scores_rating), "NA_cate"))
mv_nb$review_scores_rating[is.na(mv_nb$review_scores_rating)] <- "NA_cate"
mv_nb$host_acceptance_rate[mv_nb$host_acceptance_rate == "N/A"] <- 0
mv_nb$host_acceptance_rate <- as.numeric(gsub("[^0-9.]", "", mv_nb$host_acceptance_rate))
mv_nb$host_acceptance_rate<-mv_nb$host_acceptance_rate/100
breaks_host <- quantile(mv_nb$host_acceptance_rate, probs = seq(0, 1, 1/2), na.rm = TRUE)
mv_nb$host_acceptance_rate<-cut(mv_nb$host_acceptance_rate,breaks=breaks_host,
                                labels = c("low","high"),
                                include.lowest = TRUE)
mv_nb$host_is_superhost<-as.factor(mv_nb$host_is_superhost)
mv_nb$host_response_time<-as.factor(mv_nb$host_response_time)
mv_nb$room_type<-as.factor(mv_nb$room_type)
set.seed(699)
train_prop<-0.6
train.index<-sample(c(1:nrow(mv_nb)),nrow(mv_nb)*train_prop)
train.fit<-mv_nb[train.index,]
valid.fit<-mv_nb[-train.index,]
fitnb<-naiveBayes(review_scores_rating~.,data=train.fit)
fitnb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##       low       mid      high   NA_cate
## 0.2821429 0.2869048 0.2845238 0.1464286
##
## Conditional probabilities:
##          host_is_superhost
## Y              FALSE       TRUE
##   low     0.90295359 0.09704641
##   mid     0.46473029 0.53526971
##   high    0.57740586 0.42259414
##   NA_cate 0.91056911 0.08943089
##
##          host_response_time
## Y         a few days or more         N/A within a day within a few hours
##   low            0.067510549 0.084388186  0.092827004        0.101265823
##   mid            0.004149378 0.120331950  0.066390041        0.058091286
##   high           0.029288703 0.129707113  0.058577406        0.133891213
##   NA_cate        0.097560976 0.260162602  0.081300813        0.138211382
##          host_response_time
## Y         within an hour
##   low        0.654008439
##   mid        0.751037344
##   high       0.648535565
##   NA_cate    0.422764228
##
##          host_acceptance_rate
## Y               low      high
##   low     0.5991561 0.4008439
##   mid     0.4481328 0.5518672
##   high    0.5648536 0.4351464
##   NA_cate 0.6910569 0.3089431
##
##          room_type
## Y         Entire home/apt  Hotel room Private room Shared room
##   low         0.721518987 0.021097046  0.253164557 0.004219409
##   mid         0.775933610 0.004149378  0.219917012 0.000000000
##   high        0.794979079 0.000000000  0.196652720 0.008368201
##   NA_cate     0.691056911 0.016260163  0.268292683 0.024390244
```

In order to make a Naïve Bayes model to predict the review score rating for each house, three equal frequency bins were applied to review_score_rating variables to assign three groups for the variable, which were low, mid, and high that represent low review rating, mid-level rating, and high rating. Then, features such as whether the host is a super host, time the host take a response, the acceptance rate, and the room type were chosen to predict

the review score rating because a host service to customers can play a important role in affecting customers rating to the house. It is also important that the type of the house can affect the living experience of customers. Therefore, these variables can help in determining the level of review rating for Airbnb houses.

Step 3 Part 2 Task B

```
fakehouse<-data.frame(
  host_is_superhost="TRUE",
  host_response_time="within an hour",
  host_acceptance_rate="high",
  room_type="Private room"
)
prediction <- predict(fitnb, fakehouse)
prediction
```

```
## [1] mid
## Levels: low mid high NA_cate
```

The fictional person is a superhost with a response time within an hour, high acceptance rate, and listing private room. The model predict that this person will receive mid-level review score.

Step 3 Part 2 Task C

```
fitnb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##       low       mid      high   NA_cate
## 0.2821429 0.2869048 0.2845238 0.1464286
##
## Conditional probabilities:
##          host_is_superhost
## Y              FALSE       TRUE
##   low     0.90295359 0.09704641
##   mid     0.46473029 0.53526971
##   high    0.57740586 0.42259414
##   NA_cate 0.91056911 0.08943089
##
##          host_response_time
## Y         a few days or more         N/A within a day within a few hours
##   low            0.067510549 0.084388186  0.092827004        0.101265823
##   mid            0.004149378 0.120331950  0.066390041        0.058091286
##   high           0.029288703 0.129707113  0.058577406        0.133891213
##   NA_cate        0.097560976 0.260162602  0.081300813        0.138211382
##          host_response_time
## Y         within an hour
##   low        0.654008439
##   mid        0.751037344
##   high       0.648535565
##   NA_cate    0.422764228
##
##          host_acceptance_rate
## Y               low      high
##   low     0.5991561 0.4008439
##   mid     0.4481328 0.5518672
##   high    0.5648536 0.4351464
##   NA_cate 0.6910569 0.3089431
##
##          room_type
## Y         Entire home/apt  Hotel room Private room Shared room
##   low         0.721518987 0.021097046  0.253164557 0.004219409
##   mid         0.775933610 0.004149378  0.219917012 0.000000000
##   high        0.794979079 0.000000000  0.196652720 0.008368201
##   NA_cate     0.691056911 0.016260163  0.268292683 0.024390244
```

```
train_predictions <- predict(fitnb, train.fit)
validation_predictions <- predict(fitnb, valid.fit)
train_cm <- table(Predicted = train_predictions, Actual = train.fit$review_scores_rating)
validation_cm <- table(Predicted = validation_predictions, Actual = valid.fit$review_scores_rati
ng)
train_accuracy <- sum(diag(train_cm)) / sum(train_cm)
validation_accuracy <- sum(diag(validation_cm)) / sum(validation_cm)
```

```
train_accuracy
```

```
## [1] 0.427381
```

```
validation_accuracy
```

```
## [1] 0.4267857
```

S III II D: After the model is conducted and tested, the performance was assessed by accuracy rate. Based on the model performance, the train set had an accuracy rate of 0.4274 while the validation set had an accuracy rate of 0.4268. The difference between in accuracy rate between train set and validation set is similar, which suggests that there are no potential overfitting problems.

It is also important that since the outcome variable is mostly evenly distributed, a naive model that always predicts the most common class would have an accuracy of about 25%. In this case, a 43% accuracy would be significantly better than the baseline.

Step 3 Part 3

```
#tree model
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.2.3
```

```
library(rpart.plot)
```

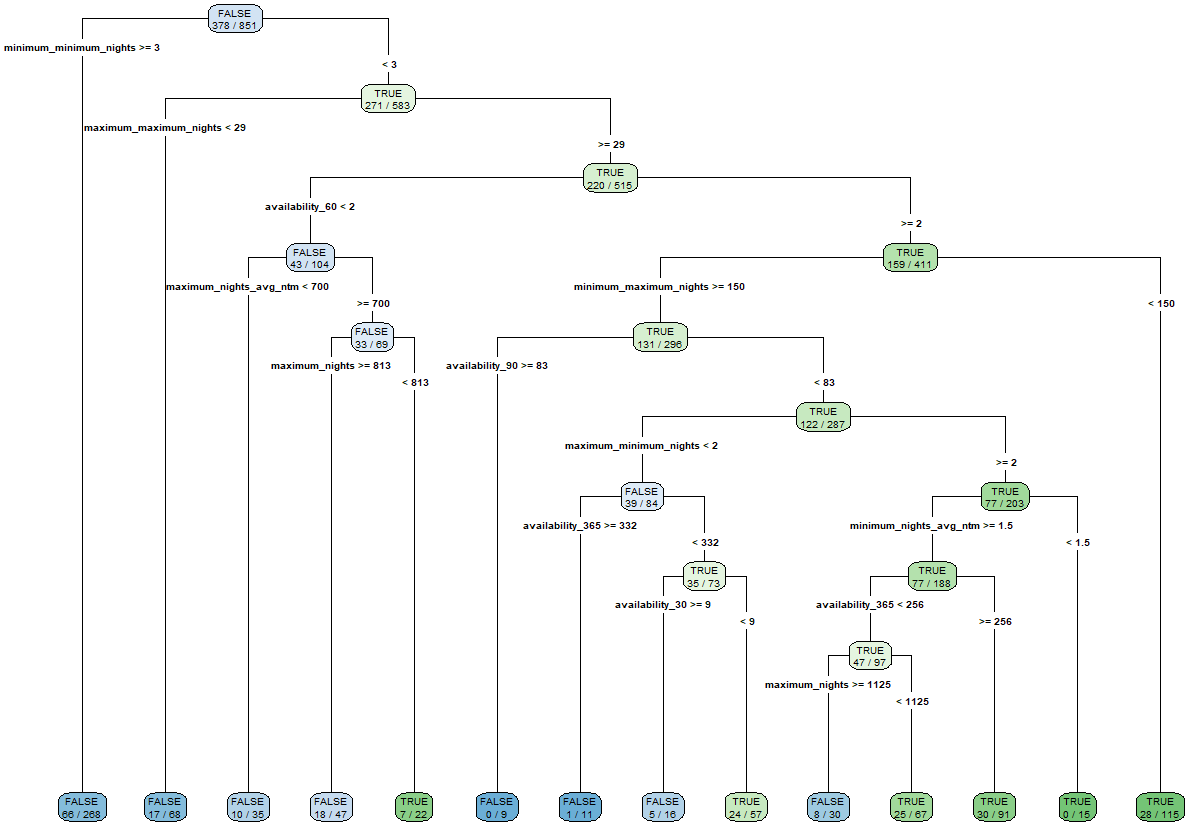
```
## Warning: package 'rpart.plot' was built under R version 4.2.3
```

```
tree_df <- mv%>% select('instant_bookable', 42:49, 52:55)

tree_df$instant_bookable <- as.factor(tree_df$instant_bookable)

set.seed(699)
train.index <- sample(c(1:nrow(tree_df)), nrow(tree_df)*0.6)
train.df <- tree_df[train.index, ]
valid.df <- tree_df[-train.index, ]

tree <- rpart(instant_bookable~., data=train.df, method= 'class')
rpart.plot(tree, type = 4, extra = 3)
```



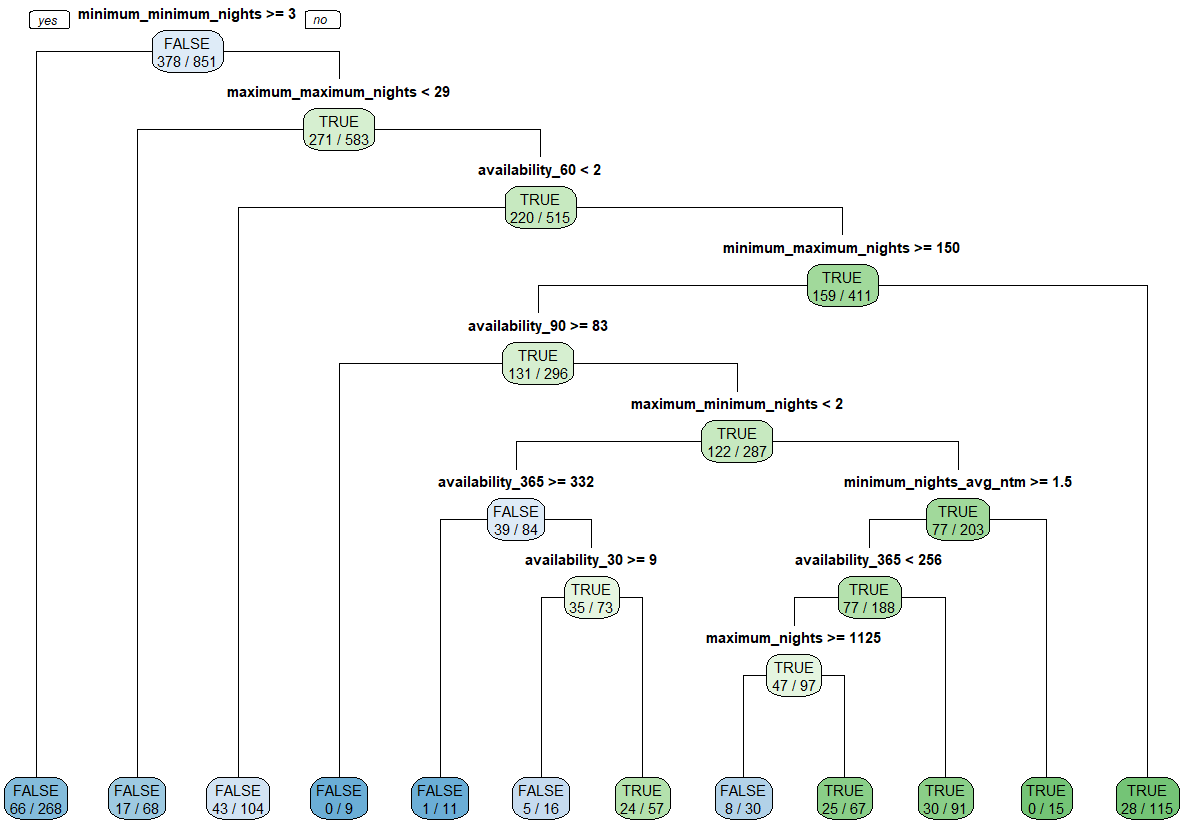

```
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = instant_bookable ~ ., data = train.df, method = "class")
##
## Variables actually used in tree construction:
##  [1] availability_30        availability_365       availability_60
##  [4] availability_90        maximum_maximum_nights maximum_minimum_nights
##  [7] maximum_nights         maximum_nights_avg_ntm minimum_maximum_nights
## [10] minimum_minimum_nights minimum_nights_avg_ntm
##
## Root node error: 378/851 = 0.44418
##
## n= 851
##
##         CP nsplit rel error  xerror     xstd
## 1 0.108466      0   1.00000 1.00000 0.038346
## 2 0.089947      1   0.89153 1.02910 0.038445
## 3 0.047619      2   0.80159 0.84127 0.037335
## 4 0.011905      3   0.75397 0.81481 0.037087
## 5 0.010582     11   0.65344 0.80423 0.036981
## 6 0.010000     13   0.63228 0.82275 0.037164
```

Classification, Part III B: Based on the cp table above, we decide to implement cp of 0.010582 with minsplit of 11 since it produces the lowest x error here. Classification, Part III C: Here is the graph of our tree.

```
new_tree <- rpart(instant_bookable~., data=train.df, method = 'class',
                  cp=0.010582, minsplit=11)

rpart.plot(new_tree, type = 1, extra = 3)
```




```
prediction_of_my_tree_train <- predict(new_tree, train.df, type='class')
cf_tree_train <- confusionMatrix(prediction_of_my_tree_train, train.df$instant_bookable, positiv
e = 'TRUE')
print(cf_tree_train)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   366  140
##      TRUE    107  238
##
##                Accuracy : 0.7098
##                  95% CI : (0.678, 0.7401)
##     No Information Rate : 0.5558
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.407
##
##  Mcnemar's Test P-Value : 0.04174
##
##             Sensitivity : 0.6296
##             Specificity : 0.7738
##          Pos Pred Value : 0.6899
##          Neg Pred Value : 0.7233
##              Prevalence : 0.4442
##          Detection Rate : 0.2797
##    Detection Prevalence : 0.4054
##       Balanced Accuracy : 0.7017
##
##        'Positive' Class : TRUE
##
```

```
prediction_of_my_tree_valid <- predict(new_tree, valid.df, type='class')
cf_tree_valid <- confusionMatrix(prediction_of_my_tree_valid, valid.df$instant_bookable, positiv
e = 'TRUE')
print(cf_tree_valid)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   233  103
##      TRUE     97  135
##
##                Accuracy : 0.6479
##                  95% CI : (0.607, 0.6872)
##     No Information Rate : 0.581
##     P-Value [Acc > NIR] : 0.0006521
##
##                   Kappa : 0.2743
##
##  Mcnemar's Test P-Value : 0.7236736
##
##             Sensitivity : 0.5672
##             Specificity : 0.7061
##          Pos Pred Value : 0.5819
##          Neg Pred Value : 0.6935
##              Prevalence : 0.4190
##          Detection Rate : 0.2377
##    Detection Prevalence : 0.4085
##       Balanced Accuracy : 0.6366
##
##        'Positive' Class : TRUE
##
```

Classification, Part III D: We start our modeling process by searching for inputs that might seem logic in predicting whether it is instantly bookable or not. We decide to take all the variables that are related to availability and nights as inputs here. After cross-validation, we implement those best hyper parameters for the tree model and refit the data as needed. The tree automatically locates the most important feature here which is minimum minimum nights based on the graph above. According to the confusion matrix above, the model has an accuracy of 0.65 and TPR rate of 0.57 which is relative feasible for predicting.

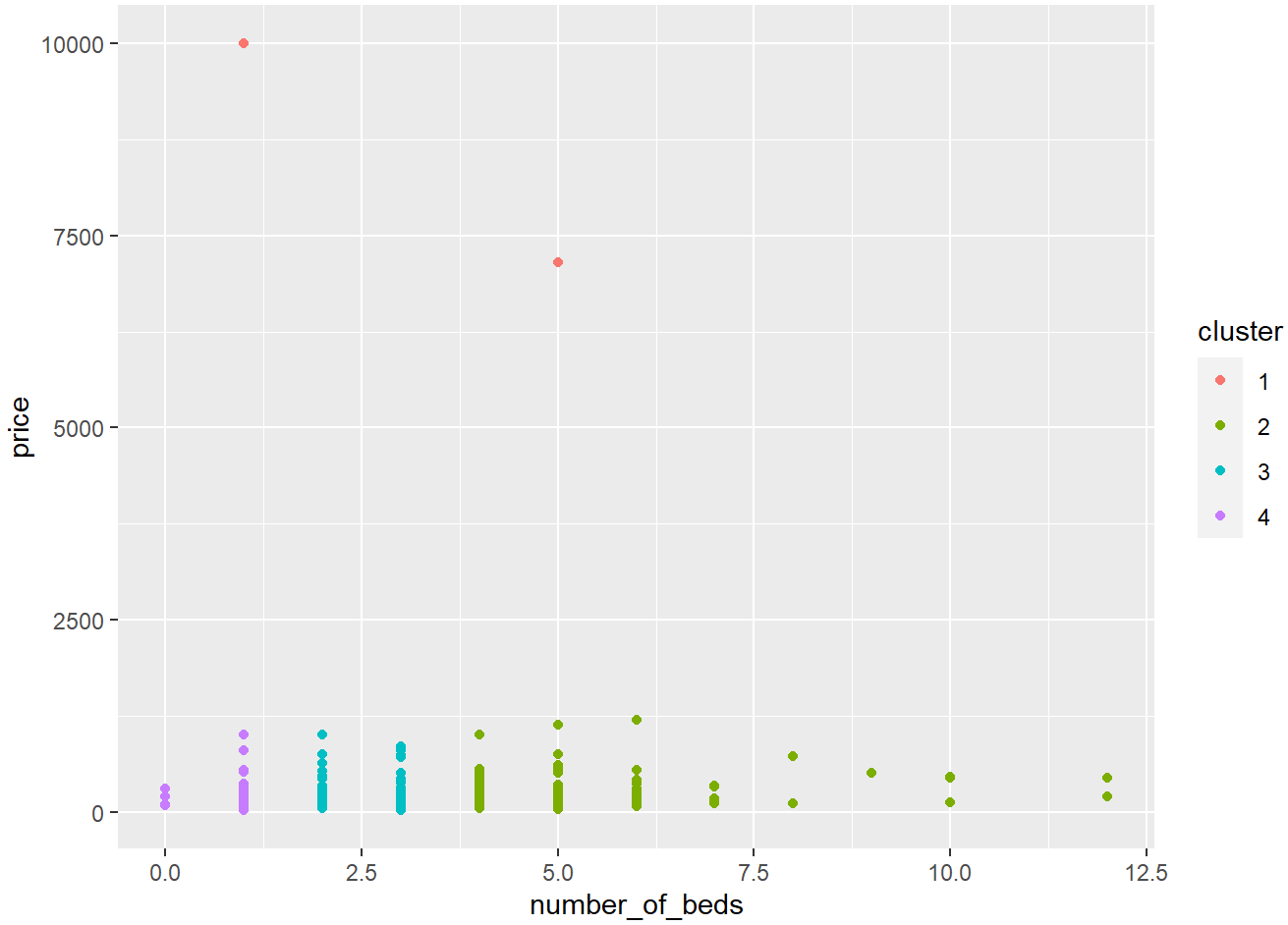
Step 4 Cluster Step 4 Task A and B

```
mv_cluster<- select(mv,id,number_of_beds,price)
sum(is.na(mv_cluster))
```

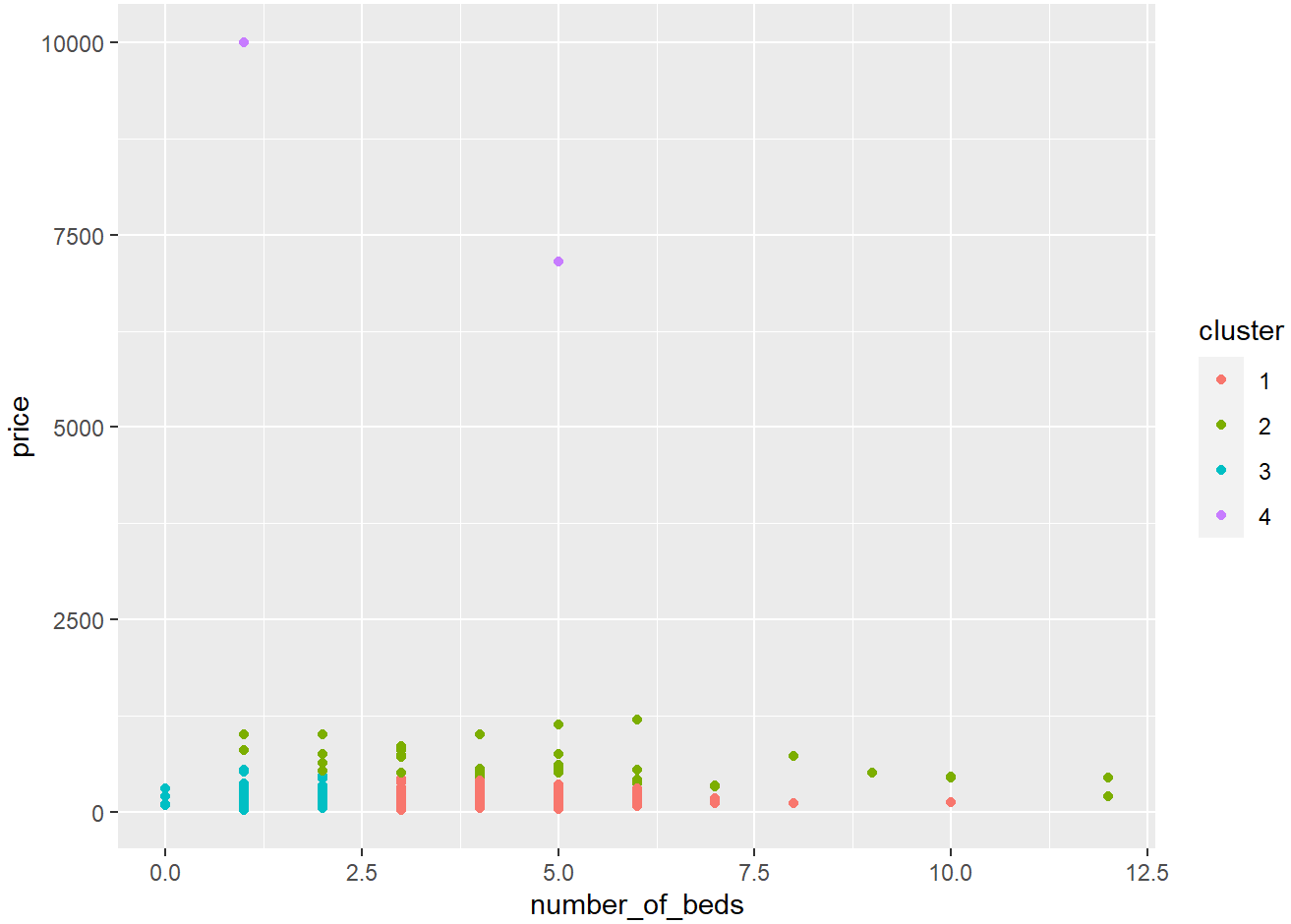
```
## [1] 0
```

```
mv_cluster$price <- as.numeric(gsub("[\\$,]", "", mv_cluster$price))
```

```
mv_cluster$id<-as.character(mv_cluster$id)
mv_cluster<-as.data.frame(mv_cluster)
row.names(mv_cluster)<-mv_cluster[,1]
mv_cluster<-mv_cluster[,-1]
mv_cluster_norm<-sapply(mv_cluster,scale)
d.norm<-dist(mv_cluster_norm,method="euclidean")
km<-kmeans(mv_cluster_norm,4)
mv_cluster$cluster <- as.factor(km$cluster)
ggplot(mv_cluster,aes(x=number_of_beds,y=price,color=cluster))+geom_point()
```




```
set.seed(699)
mv_cluster_norm<-as.data.frame(mv_cluster_norm)
mv_cluster_norm$price<-mv_cluster_norm$price*2
d.norm<-dist(mv_cluster_norm,method="euclidean")
km<-kmeans(mv_cluster_norm,4)
mv_cluster$cluster <- as.factor(km$cluster)
ggplot(mv_cluster,aes(x=number_of_beds,y=price,color=cluster))+geom_point()
```




```
mv_cluster$cluster<- ifelse(mv_cluster$cluster == 1, "More Bed Normal Price",
                            ifelse(mv_cluster$cluster == 2, "High End House",
                                   ifelse(mv_cluster$cluster == 3, "Normal Bed Normal Price","Luxu
ry")))
```

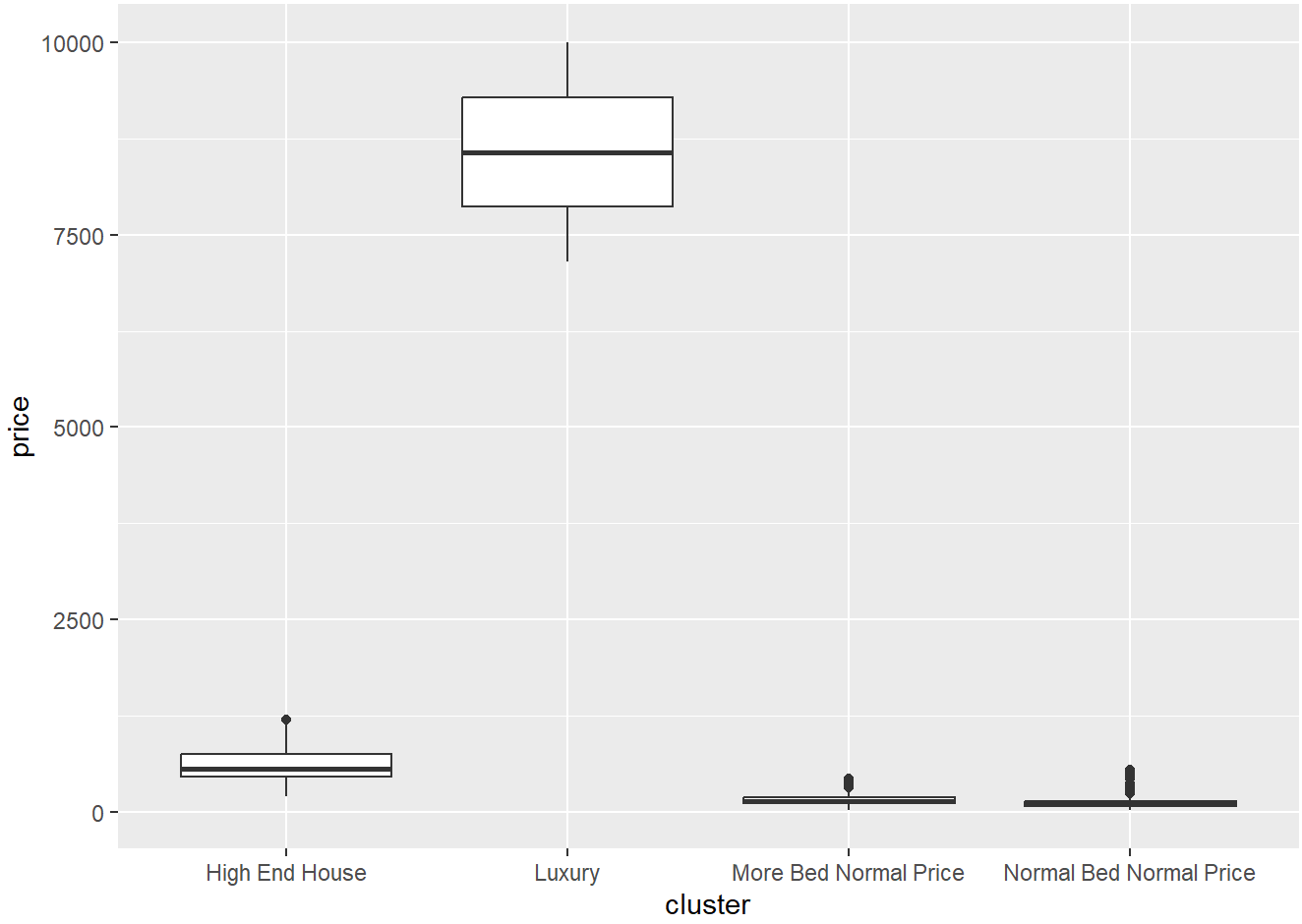
For cluster analysis, two variables, number of bed and price, were chosen to cluster all Airbnb houses to 4 different clusters. The reason why these two variables were chosen is that for a customer who wants to book a house, the two main factors that affect decision making are how many people the house can serve and how much price they need to pay. Thus, the number of beds and price of the house are the essential variables that affect customer's decision making and were chosen to differentiate clusters.

The method of clustering used is k-means cluster. Since values of price variable are much higher than number of beds, normalization of variables was conducted to remove the effect of big numbers. The initial clustering was not successful because 4 different clusters are mainly based on number of beds rather than both beds and prices. Therefore, the weight of price was doubled to find out if there is any change. The result of double weighted price shows a good result that distinguish different clusters by both beds and price.

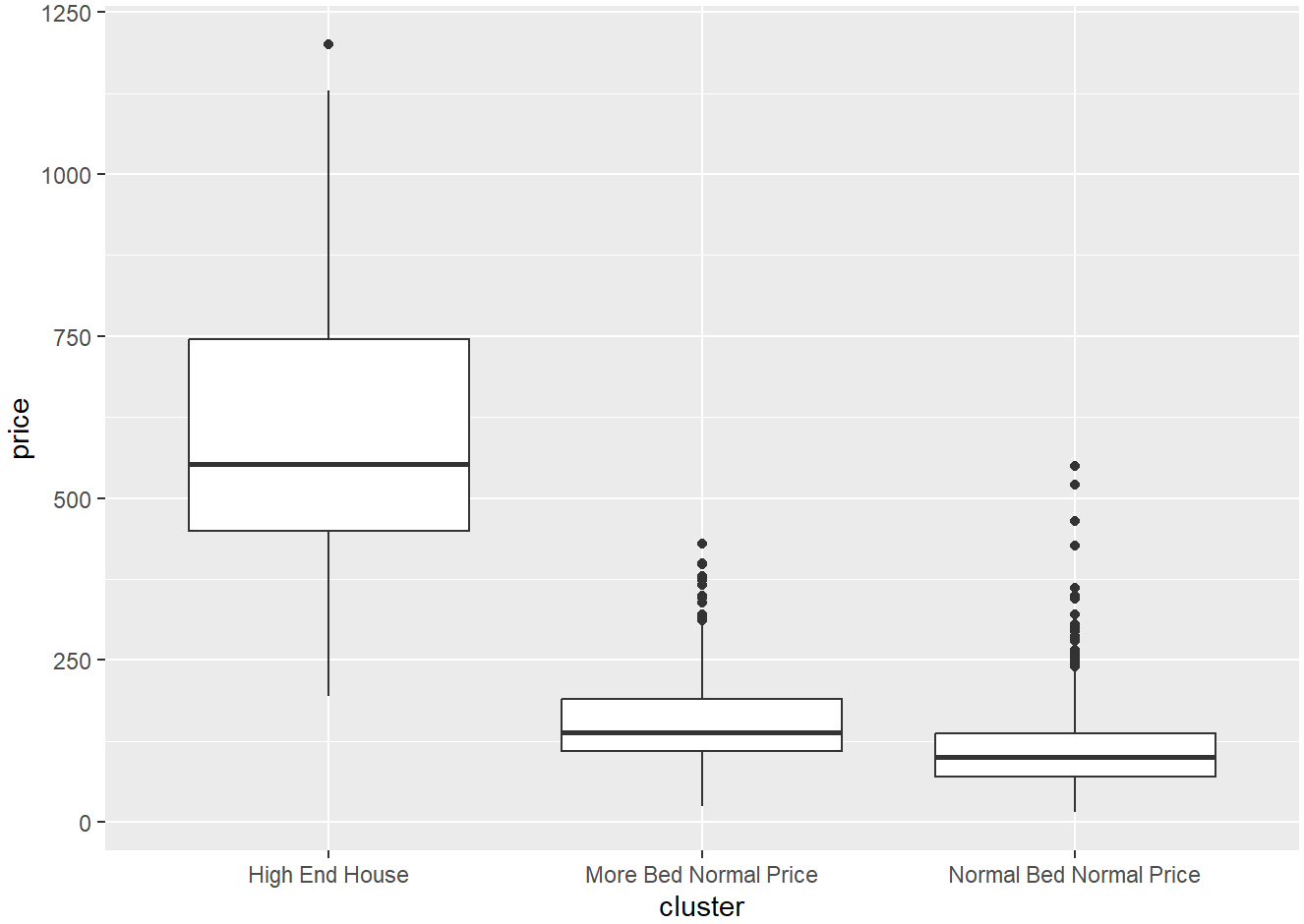
The four different clusters were given the following names, high end house, luxury, more bed normal price, and normal bed normal price. High end house represents houses that have higher prices. Luxury represents houses that charge extremely high prices for just 1 night. More bed normal price represents houses that have more than 3 beds with normal price. Normal bed normal price represents houses that have about 0 to 2 beds with normal price.

Step 4 Part 3

```
ggplot(mv_cluster,aes(x=cluster,y=price))+geom_boxplot()
```

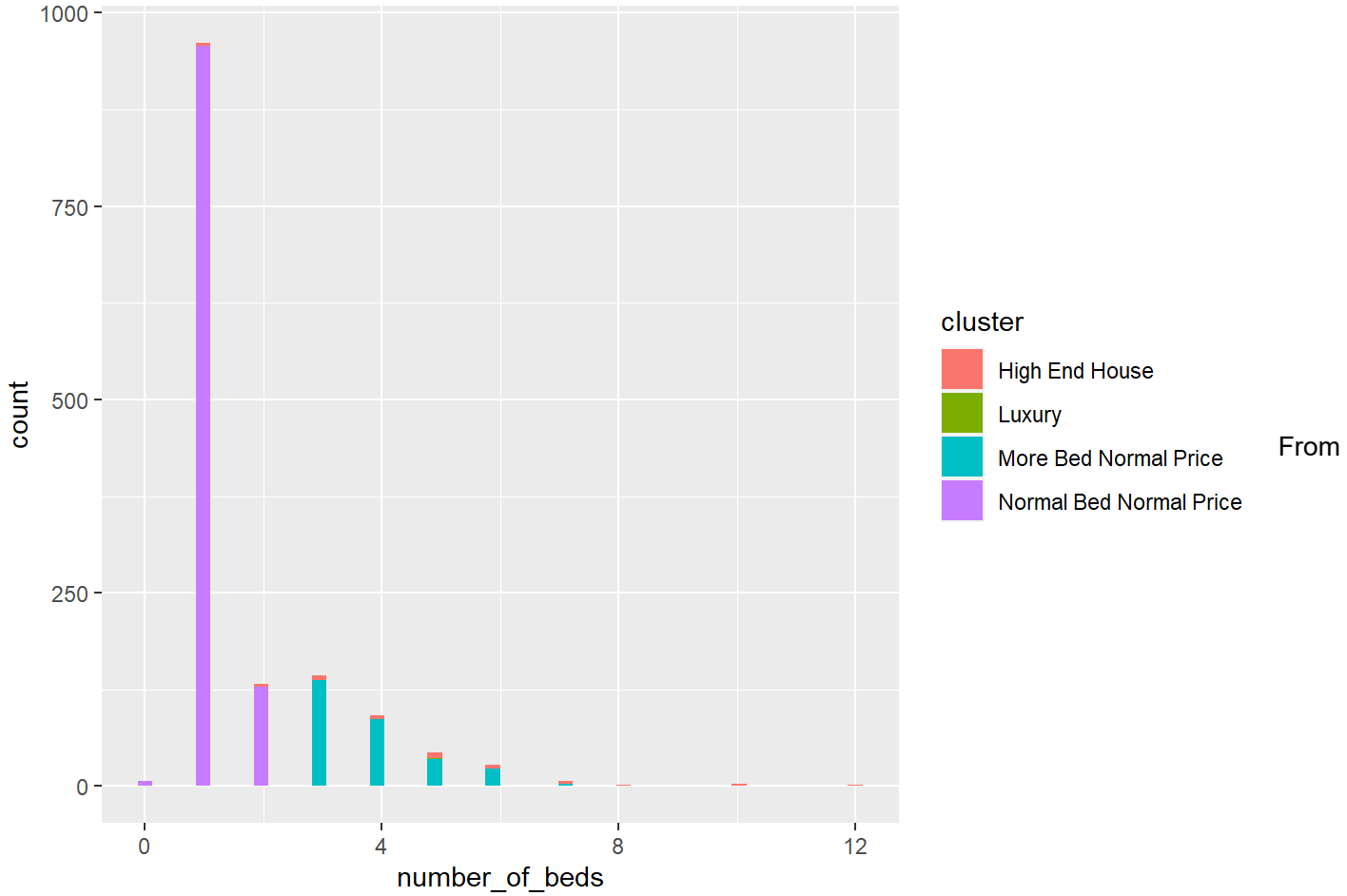



```
mv_cluster_nol<-mv_cluster %>% filter(cluster !="Luxury")
ggplot(mv_cluster_nol,aes(x=cluster,y=price))+geom_boxplot()
```



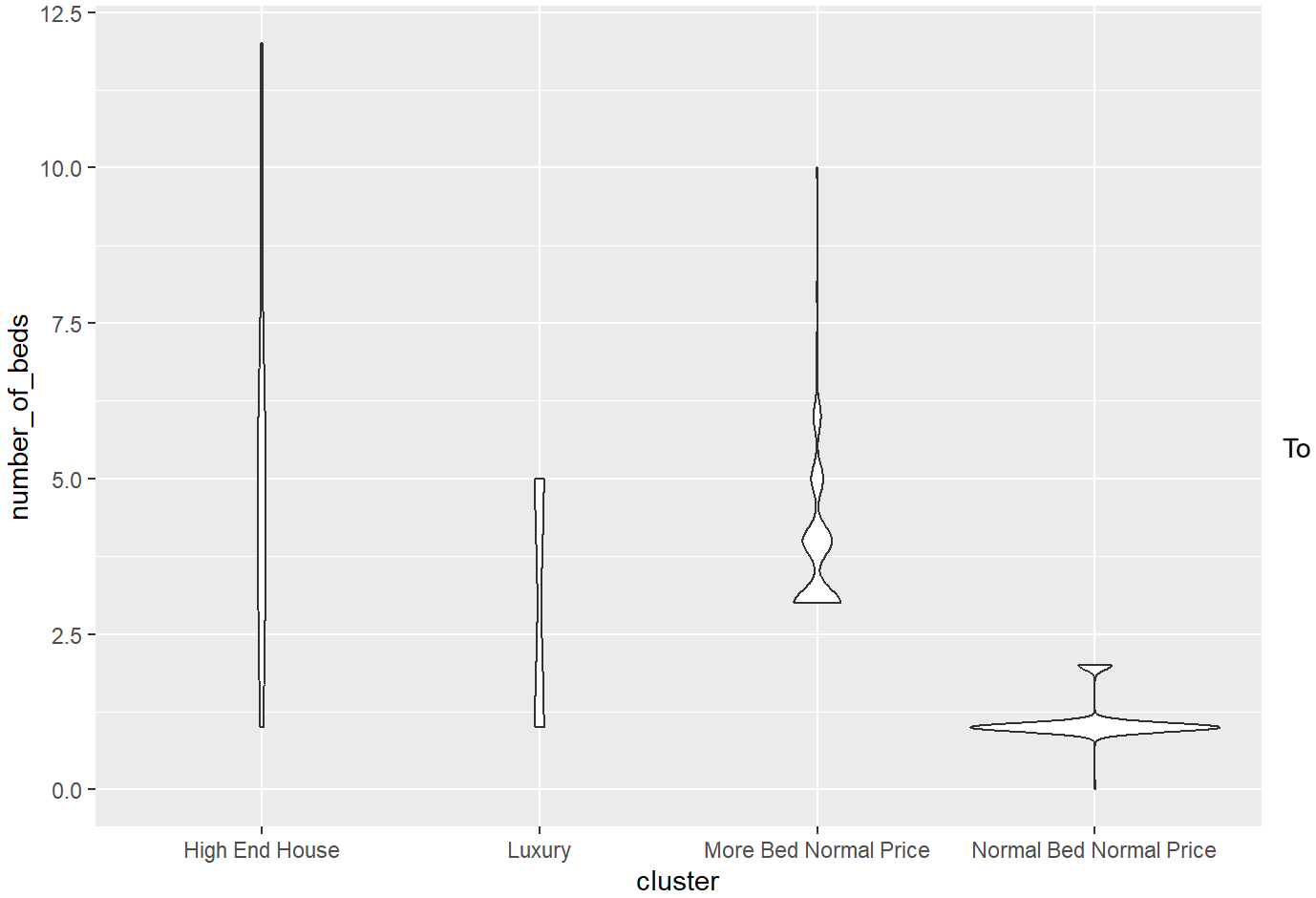
From the box plot of prices, it is noticeable that Luxury house charges extremely high prices that even above $7500 per night, while other houses charge a lower price. By removing the outliers from the luxury house, high end house typically charges price that is about $500, while the remaining three clusters charge prices range from $100 to $200.

```
ggplot(mv_cluster,aes(x=number_of_beds,fill=cluster))+geom_histogram(bins=50)
```



From the histogram of beds, it is obvious that Normal number bed houses have number of beds around 0-3. More number bed houses have number of beds more than 3. For Luxury houses, number of beds seems not affect the price. The number of beds range from 2 to 5. For high end houses, the number of beds range from 1 to over 10.

```
ggplot(mv_cluster,aes(x=cluster,y=number_of_beds))+geom_violin()
```



have a clearer look about the number of beds shown in the data set, a violin plot is also generated. It can be found that for high end houses, the number of beds range from 1 to 12, while other clusters demonstrate the similar outcomes compared to the histogram.

Step V Conclusion:

The data preparation and exploration process is crucial for the rest of the project. It is good to check what each column contains, their data type, and if there are any redundancy or inter-dependency. For example, the URL columns will not help with data analysis and there are many columns that all contain the bedroom/bed number information in our case, which can be taken out to make the future process easier. For property rental companies like Airbnb, it is also important to reflect on whether tracking certain metrics are necessary, since it does cost resource to do so and might turn out to be a waste.

Many advertising/travel agency companies can really take advantage of the clustering models to feed customers more valuable information. For example, whenever there is a new customer start the vacation planning process, the agency can let them take a survey first. The survey will contain questions about key topics/metrics. Companies then use this result to place that customer into a certain cluster and start with item recommendations from that cluster. The company can also show the customer reviews from others in the same cluster, as “see those who are similar to you also says”.

This project also inspires us on how to deal with missing values within the data set in the future. The Naive Bayes classification part is a great example. The company can use other historical data to simply “predicting” the missing value as a alternative solution to using mean/median/mode. On top of that, the company can also change the target variable from a specific number to a categorical range in order to improve its accuracy by using this method. For example, instead of predicating a numeric value for the price, they can divide the result into low, medium, and high and give one of these three as result.