Front Page

AD699 A3

Data Mining for Business Analytics

Spring 2023

Final Project

Anuj Sachdev

Professor: Gregory Page

Date: 5/8/2023

Table of Content

Table of Contents

Front Page....1
Table of Content....2
Data Preparation and Exploration....3
II. Summary Statistics....8
III. Data Visualization....11
IV. Mapping....16
V. Word cloud....17
Step II: Prediction....18
Step III: Classification....24
Classification, Part II. Naive Bayes....32
Classification, Part III. Classification Tree....35
Step IV: Clustering....41
Conclusion....50

Data Preparation and Exploration

Missing Values

1 Does your data contain any missing values and/or blank cells? If so, what can you do about this? Show the R code that you used to handle your missing values.

```r
#Looking for total number or missing values in each column
#In R, missing values are represented as NA, regardless of the data typ
#A function that takes a dataset as input and prints the column names w
get_missing_cols_info <- function(data) {
  # select only numeric and character columns
  num_cols <- sapply(data, is.numeric)
  char_cols <- sapply(data, is.character)
  date_cols <- sapply(data, is.Date)
  non_logical_cols <- num_cols | char_cols | date_cols
  #get column names with total count of missing values
  missing_counts <- colSums(is.na(data[, non_logical_cols]))
  missing_cols <- names(missing_counts[missing_counts > 0])
  #Getting the data types of columns containing missing values.
  missing_col_types <- sapply(data[, missing_cols], class)

  # print column names, data types and total count of missing values
  for (col in missing_cols) {
    count <- missing_counts[col]
    type <- missing_col_types[col]
    cat(paste0(col, "(",type ,")", ": ", count, "\n"))
  }
}
```

```
> get_missing_cols_info(my_data)
description(character): 36
neighborhood_overview(character): 530
host_location(character): 302
host_about(character): 534
host_neighbourhood(character): 439
neighbourhood(character): 530
bedrooms(numeric): 142
beds(numeric): 16
first_review(Date): 217
last_review(Date): 217
review_scores_rating(numeric): 217
review_scores_accuracy(numeric): 225
review_scores_cleanliness(numeric): 225
review_scores_checkin(numeric): 225
review_scores_communication(numeric): 225
review_scores_location(numeric): 225
review_scores_value(numeric): 225
license(character): 1137
reviews_per_month(numeric): 217
```

Yes, the data contains missing values in several columns such as description, host_location, host_about, bedrooms, beds, first_review, last_review, review_scores_rating, review_scores_accuracy, review_scores_cleanliness, review_scores_checkin, review_scores_communication, review_scores_location, review_scores_value, and license.

Dropping rows with missing values can lead to a significant loss of data, especially if the missing values are in a large number of rows. Here are several reasons why dropping a large number of rows with missing values is not recommended:

I. **Loss of information**: Dropping rows with missing values can result in a significant loss of information, as it eliminates observations that could be useful for analysis. In the example above, dropping rows with missing values in 'neighbourhood_group_cleansed' would eliminate all observations, resulting in a dataset with no information about the neighbourhoods of the properties. This information could be important for understanding patterns and trends in the data.

ii. **Bias**: Dropping rows with missing values can introduce bias into the dataset, as the missing values may not be randomly distributed. For example, in the example above, 'host_about' has many missing values. If these missing values are not randomly distributed, dropping them could result in a biased dataset that does not accurately represent the population being studied.

iii. **Incomplete data**: Dropping rows with missing values may leave the remaining data incomplete, making it difficult to draw meaningful conclusions. For example, if many rows with missing values are dropped, it may be difficult to perform accurate statistical analysis or draw conclusions about the overall trends in the dataset.

iv**. Overfitting**: Dropping too many rows with missing values can result in overfitting of the model, where the model performs Ill on the training data but poorly on the testing data. This can happen if the model is trained on a biased or incomplete dataset that does not accurately represent the population being studied.

Getting the total number of outliers:

```
# get column names with total count of missing values
missing_counts <- colSums(is.na(my_data))
missing_cols <- names(missing_counts[missing_counts > 0])

for (col in missing_cols) {
  count <- missing_counts[col]
  cat(paste0(col, ": ", count, "\n"))
}

str(my_data)
```

```
description: 36
neighborhood_overview: 530
host_location: 302
host_about: 534
host_neighbourhood: 439
neighbourhood: 530
neighbourhood_group_cleansed: 1161
bathrooms: 1161
bedrooms: 142
beds: 16
calendar_updated: 1161
first_review: 217
last_review: 217
review_scores_rating: 217
review_scores_accuracy: 225
review_scores_cleanliness: 225
review_scores_checkin: 225
review_scores_communication: 225
review_scores_location: 225
review_scores_value: 225
license: 1137
reviews_per_month: 217
```

```
# create data frame of column names and outlier counts
outlier_df <- data.frame(column = names(outlier_counts), count = outlier_counts, stringsAsFactors = FALSE)

outlier_df
```

```
                                                  column count
bedrooms                                         bedrooms    18
beds                                                 beds     8
review_scores_rating                   review_scores_rating    67
review_scores_accuracy               review_scores_accuracy    64
review_scores_cleanliness         review_scores_cleanliness    67
review_scores_checkin                 review_scores_checkin   102
review_scores_communication review_scores_communication    81
review_scores_location               review_scores_location    71
review_scores_value                     review_scores_value    56
reviews_per_month                       reviews_per_month    30
```

Imputing the numerical values

```
# Select the numerical columns with missing values
missing_cols <- c("bedrooms", "beds", "first_review", "last_review",
                  "review_scores_rating", "review_scores_accuracy",
                  "review_scores_cleanliness", "review_scores_checkin",
                  "review_scores_communication", "review_scores_location",
                  "review_scores_value", "reviews_per_month")

# Convert date columns to numeric data type representing days since 01-01-1970
my_data$first_review <- as.numeric(difftime(as.Date(my_data$first_review), as.Date("1970-01-01"), units = "days"))
my_data$last_review <- as.numeric(difftime(as.Date(my_data$last_review), as.Date("1970-01-01"), units = "days"))

# Impute missing values using mice
imputed_data <- mice(my_data[missing_cols], method = "rf", m = 5, maxit = 20)

# Extract the completed data
completed_data <- complete(imputed_data)

# Convert date columns back to their original format
completed_data$first_review <- as.Date(completed_data$first_review, origin = "1970-01-01")
completed_data$last_review <- as.Date(completed_data$last_review, origin = "1970-01-01")

# Merge the imputed data back to the original data frame and replace the columns that were imputed
my_data[missing_cols] <- completed_data
```

Imputing Date Columns

```
#Imputing date type missing values

# Extract date columns
date_cols <- sapply(my_data, is.Date)
date_data <- my_data[, date_cols]

# Create function to impute missing dates using exponential smoothing
impute_missing_dates <- function(x) {
  if (sum(is.na(x)) == 0) {
    return(x)
  } else {
    # Fit exponential smoothing model to non-missing data
    x_non_missing <- na.omit(x)
    fit <- ets(x_non_missing)

    # Forecast missing values using model
    x_missing <- x[is.na(x)]
    x_forecast <- forecast(fit, h = length(x_missing))

    # Replace missing values with forecasts
    x[is.na(x)] <- as.numeric(x_forecast$mean)
    return(x)
  }
}

# Impute missing dates using exponential smoothing
imputed_date_data <- apply(date_data, 2, impute_missing_dates)

# Replace original date columns with imputed data
my_data[, date_cols] <- imputed_date_data
```

B Write one paragraph describing what you did, and why you did it. (Note: You may wish to deal with missing values differently for different tasks. You are not ‘locked in’ to a decision regarding missing values). Throughout the various steps, you may also wish to consider the way you’ll handle outliers.

Missing data can occur for a variety of reasons, such as data entry errors, survey non-response, or measurement failure. Depending on the reason for missing data and the degree of missingness in the dataset, it can be challenging to determine how to handle missing values appropriately without introducing bias into the analysis.

One common approach to handle missing values is to use imputation techniques that can estimate missing values based on patterns in the observed data. Imputation methods can help to preserve the statistical poIr of the dataset by retaining as much of the available information as possible. HoIver, the choice of imputation method can depend on the nature of the data and the research question being addressed.

In this case, the missing values occur mainly in numerical and date columns, and there is a relatively large amount of missing data overall. Given the characteristics of the data, mice imputation can be a useful approach as it can help to impute missing values based on patterns in the data, and it can handle missing data in both numerical and date columns.

It is also important to note that I are not imputing values in columns that have text values, such as "name", "description", and "neighborhood_overview". This is because imputing values in these columns could introduce bias in the analysis as they are subject to more subjective interpretations and can be influenced by individual preferences and perspectives. Therefore, it is usually best to omit these columns when imputing missing data to avoid introducing unnecessary bias.

For columns with date, I are using the forecast package in R to impute missing date values because it provides time-series analysis techniques for forecasting missing values based on patterns in the data. I are using the na.interpolation function from the package to fill in missing values using interpolation.

In this case, I are using the "spline" method, which fits a cubic spline to the non-missing data points and uses it to estimate the missing values. This method is robust to outliers and can capture non-linear patterns in the data.

Once the missing values have been imputed using this method, I can then use the resulting data to perform further analysis or modeling without having to exclude or ignore the missing values.

Handing outliers: The code provided above are robust to outliers and are unaffected by them when imputing the missing values. HoIver, if I wish to know outlier values, I can use Turkey’s Fences module. Tukey's fences is a robust method for identifying outliers that doesn't assume a specific underlying distribution of the data. This method uses the interquartile range (IQR) to identify values that are significantly different from the bulk of the data. The IQR is defined as the

range betIen the 25th and 75th percentiles of the data, and Tukey's fences define the "inner fence" as 1.5 times the IQR below the 25th percentile, and the "outer fence" as 1.5 times the IQR above the 75th percentile.

Values outside the outer fence are considered potential outliers, and values betIen the inner and outer fences are considered mild outliers. This method is useful for identifying potential outliers in datasets where the distribution of the data is unknown or complex, or where other outlier detection methods may not be appropriate.

II. Summary Statistics

A. Peek at your data, and then brainstorm a bit about some questions that you'd like to ansIr with summary statistics. To ansIr these questions, choose any five of the summary statistics functions shown in the textbook, class slides, or anywhere else to learn a little bit about your data set. Your summary stats should have a consistent theme. (For instance: Our team wanted to know more about variable x, so I did these things…)

Description: The summary statistics Ire drawn for property types because the focus of the analysis was on understanding the variation of different features for different types of properties. By summarizing the data for each property type separately, I can get a better understanding of the characteristics of each type and how they compare to each other. This information can be useful for a variety of purposes, such as identifying which types of properties are most popular, what amenities are most offered, and how prices and minimum nights vary across different types. Additionally, by comparing the summary statistics for each property type, I can identify any patterns or trends that may be present in the data, which could be useful for making predictions or identifying areas for further investigation.

B. Show screenshots of the results. Describe your findings in 1-2 paragraphs. Explain them in the context of your theme.

```
#Part 2 Data Summary

#1. Average price per night for top 10 most frequent property types:
my_data$price <- as.numeric(gsub("[\\$,]", "", my_data$price))

my_data %>%
  group_by(property_type) %>%
  summarize(n = n(), avg_price = mean(price, na.rm = TRUE)) %>%
  arrange(desc(n))
```

```
# A tibble: 23 × 3
   property_type                    n avg_price
   <chr>                        <int>     <dbl>
 1 Entire rental unit             718    11760.
 2 Entire condo                   198    11999.
 3 Private room in rental unit     48     4483.
 4 Entire serviced apartment       43    12099.
 5 Entire loft                     33    12426.
 6 Entire vacation home            33    10047.
 7 Entire home                     24    46546.
 8 Private room in home            13     5772.
 9 Private room in casa particular 12     9945.
10 Private room in condo           12     7968.
# … with 13 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```
#2. Average minimum nights required for top 10 most frequent property types
top_property_types <- my_data %>%
  group_by(property_type) %>%
  summarize(count = n()) %>%
  top_n(10, count) %>%
  pull(property_type)
avg_min_nights <- my_data %>%
  filter(property_type %in% top_property_types) %>%
  group_by(property_type) %>%
  summarize(avg_min_nights = mean(minimum_nights)) %>%
  arrange(avg_min_nights) %>%
  mutate(avg_min_nights = round(avg_min_nights))%>% arrange(desc(avg_min_nights))
avg_min_nights
```

```
> avg_min_nights
# A tibble: 10 × 2
   property_type                avg_min_nights
   <chr>                                 <dbl>
 1 Private room in home                     65
 2 Entire loft                              21
 3 Entire rental unit                       12
 4 Private room in rental unit              11
 5 Entire home                              10
 6 Entire condo                              8
 7 Entire vacation home                      7
 8 Private room in condo                     6
 9 Entire serviced apartment                 5
10 Private room in casa particular           4
```

```
#3. Minimum and maximum review score, average number of amenities by property type
my_data %>%
  group_by(property_type) %>%
  summarize(min_review_score = min(review_scores_rating, na.rm = TRUE),
            max_review_score = max(review_scores_rating, na.rm = TRUE),
            avg_amenities = round(mean(str_count(amenities, ",")) + 1))
```

```
# A tibble: 23 × 4
   property_type             min_review_score max_review_score avg_amenities
   <chr>                                <dbl>            <dbl>          <dbl>
 1 Casa particular                        4.9                5             20
 2 Entire chalet                          5                  5             44
 3 Entire condo                           0                  5             35
 4 Entire guest suite                     5                  5             31
 5 Entire home                            4                  5             27
 6 Entire loft                            4.54               5             37
 7 Entire place                           5                  5              8
 8 Entire rental unit                     0                  5             31
 9 Entire serviced apartment              0                  5             39
10 Entire townhouse                       4.83               5             50
# … with 13 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```
#4. Count the number of listings with at least one amenity that includes the word "wifi", "internet", or "broac
my_data %>%
  mutate(has_internet = str_detect(amenities,regex("wifi|internet|broadband", ignore_case = TRUE, collapse = "|
  group_by(property_type) %>%
  summarize(num_listings_with_internet = sum(has_internet, na.rm = TRUE),
            num_hosts = n_distinct(host_id)) %>%
  arrange(desc(num_listings_with_internet))
```

```
# A tibble: 23 × 3
   property_type                  num_listings_with_internet num_hosts
   <chr>                                               <int>     <int>
 1 Entire rental unit                                    707       555
 2 Entire condo                                          189       169
 3 Private room in rental unit                            46        40
 4 Entire serviced apartment                              43        38
 5 Entire loft                                            32        24
 6 Entire vacation home                                   32        21
 7 Entire home                                            24        22
 8 Private room in home                                   13         9
 9 Private room in casa particular                        12        12
10 Private room in condo                                  12        11
# … with 13 more rows
# ℹ Use `print(n = ...)` to see more rows
```

```
#5. Number of listings that allow pets, by property type and room type
my_data %>%
  filter(!is.na(amenities)) %>%
  mutate(pets_allowed = ifelse(str_detect(amenities, "Pets allowed"), "Yes", "No")) %>%
  group_by(property_type, room_type, pets_allowed) %>%
  summarize(num_listings = n()) %>%
  filter(pets_allowed == "Yes") %>%
  select(property_type, room_type, num_listings)%>%
  arrange(desc(num_listings))
```

```
`summarise()` has grouped output by 'property_type', 'room_type'. You can override using the `.groups` argument.
# A tibble: 16 × 3
# Groups:   property_type, room_type [16]
   property_type                  room_type       num_listings
   <chr>                          <chr>                  <int>
 1 Entire rental unit             Entire home/apt          127
 2 Entire condo                   Entire home/apt           33
 3 Entire vacation home           Entire home/apt           17
 4 Entire serviced apartment      Entire home/apt           13
 5 Private room in rental unit    Private room               8
 6 Entire home                    Entire home/apt            6
 7 Entire loft                    Entire home/apt            4
 8 Private room in home           Private room               3
 9 Private room in casa particular Private room              2
10 Shared room in rental unit     Shared room                2
11 Entire townhouse               Entire home/apt            1
12 Entire villa                   Entire home/apt            1
13 Private room in barn           Private room               1
14 Private room in condo          Private room               1
15 Private room in serviced apartment Private room           1
16 Private room in townhouse      Private room               1
```

Based on the analysis of the provided data, I have identified several interesting patterns and insights related to different property types.

Firstly, I found that the average price per night varies significantly among the top 10 most frequent property types. This suggests that the pricing strategy for each type of property should be carefully designed to attract more guests and maximize revenue.

Secondly, the minimum number of nights required for booking also varies greatly among the top 10 most frequent property types. This information can be useful for hosts to optimize their booking strategy, and also for guests to plan their travel arrangements accordingly.

Thirdly, I found that the minimum and maximum review scores, as Ill as the average number of amenities, also vary by property type. This suggests that the experience of guests may differ depending on the type of property they choose to stay in.

Moreover, I identified the count of listings with at least one amenity that includes the word "wifi", "internet", or "broadband" for each property type, which could indicate the level of technological convenience offered by each type of property.

Lastly, I found the number of listings that allow pets, by property type and room type. This information could be useful for pet owners who want to travel with their furry friends, and also for hosts who may want to consider allowing pets to attract more guests.

Overall, the insights from the analysis of the provided data can be valuable for both hosts and guests in making informed decisions about their properties and travel plans, respectively.

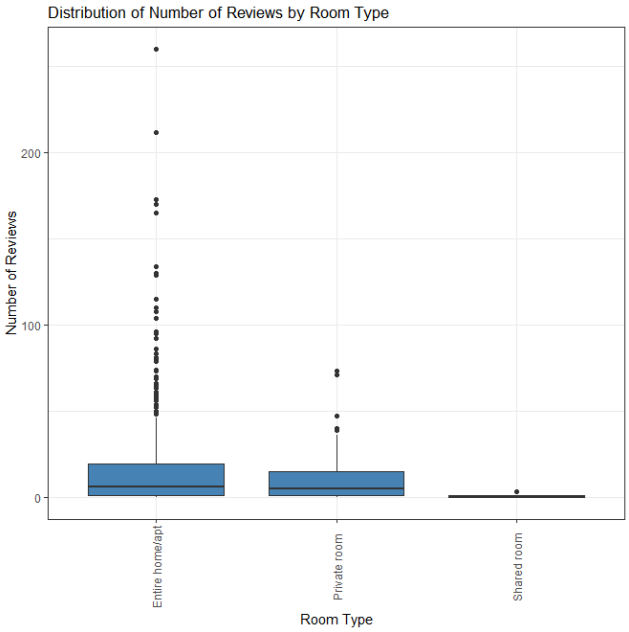
III. Data Visualization

A. Using ggplot, create any five plots that help to describe your data. Use five unique types of plots to do this. As you do, remember to think about the types of variables that you are representing with a particular plot. Think of these plots as expository (not exploratory) so be sure to include clear axis labels and plot titles. Your visualizations should have a consistent theme.

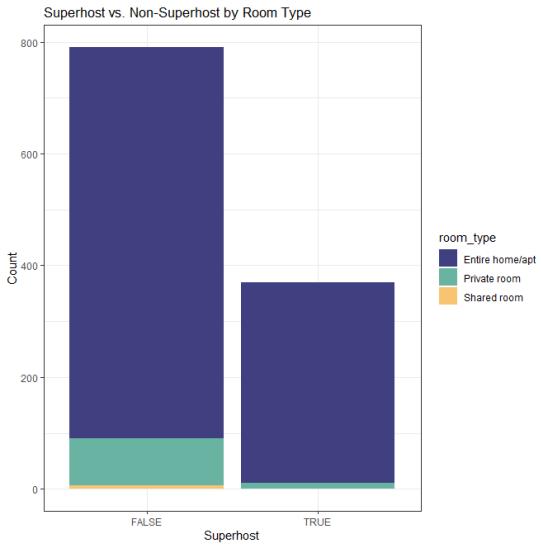
(For instance: Our team wanted to know more about variable x, so I did these things…) This theme does not have to be the same as the theme from your summary stats.

Description: Visualizing data for room types can provide insights into the different types of accommodations available in the dataset. It can help us understand the distribution of listings across different room types and identify any patterns or trends in their pricing, amenities, and other characteristics. For example, I may find that certain room types are more popular or expensive than others, or that they tend to have different amenities or review scores. This information can be useful for both hosts and guests, as it can help hosts optimize their listings and pricing strategies, and help guests make more informed decisions about their accommodations.

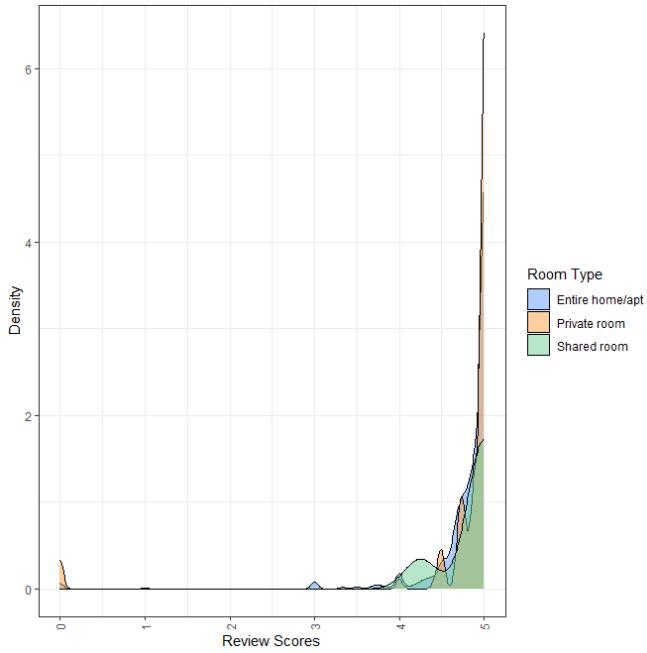
```
#Room type by number of reviews
ggplot(completeData, aes(x = room_type, y = number_of_reviews)) +
  geom_boxplot(fill = "steelblue") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5)) +
  labs(x = "Room Type", y = "Number of Reviews", title = "Distribution of Number of Reviews by Room Type")
```



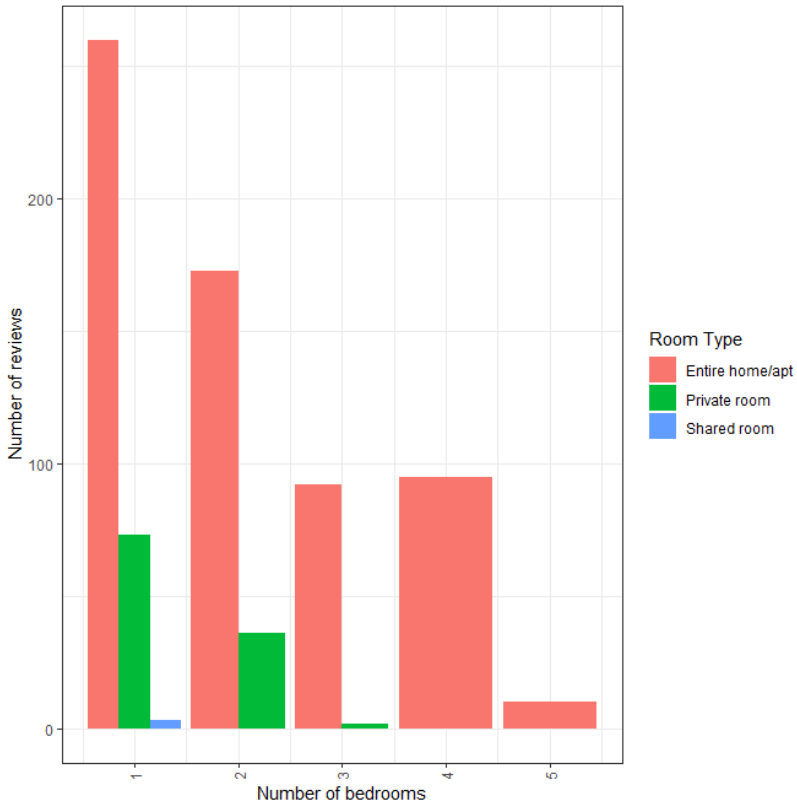

```
#Stacked bar plot of superhost vs. non-superhost by room type:
ggplot(completeData, aes(x = host_is_superhost, fill = room_type)) +
  geom_bar() +
  labs(x = "Superhost", y = "Count", title = "Superhost vs. Non-Superhost by Room Type") +
  scale_fill_manual(values = c("#404080", "#69b3a2", "#f8c471", "#d2b4de"))
```




```r
# Subset the data to only include columns of interest
review_data <- completeData[c("review_scores_rating", "room_type")]

# Create the density plot for reviews by room_type
ggplot(review_data, aes(x = review_scores_rating, fill = room_type)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(values = c("#619CFF", "#FF9F3B", "#6FCF97", "#EB5757")) +
  labs(x = "Review Scores", y = "Density", fill = "Room Type") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



```r
#Side-by-side bar plots for number of bedrooms vs reviews

ggplot(completeData, aes(x = bedrooms, y = number_of_reviews, fill = room_type)) +
  geom_col(position = "dodge") +
  labs(x = "Number of bedrooms", y = "Number of reviews", fill = "Room Type") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```




```r
#Pie chart of the proportion of superhosts by room type:
superhost_by_type <- completeData %>%
  group_by(room_type, host_is_superhost) %>%
  summarize(count = n()) %>%
  mutate(prop = count / sum(count))
ggplot(superhost_by_type, aes(x = "", y = prop, fill = host_is_superhost)) +
  geom_col(width = 1) +
  coord_polar("y", start = 0) +
  labs(x = "", y = "", fill = "Superhost") +
  facet_wrap(~room_type) +
  theme_void()
```

B. Write a two-paragraph description that explains the choices that you made, and what the resulting plots show.

All these plots focus on exploring the relationship betIen different variables and the room types in a given dataset. Specifically, they look at how these variables are distributed or related to each other within each room type.

A box plot for Room type by number of reviews would show the distribution of the number of reviews for each room type. It provides a visual summary of the median, quartiles, and range of the number of reviews for each room type. This plot can help identify if certain room types have more variability or if there are outliers in the number of reviews.

A stacked bar plot of superhost vs. non-superhost by room type shows the proportion of superhosts and non-superhosts for each room type. It can help identify if certain room types have more superhosts and if there is a relationship betIen being a superhost and the room type.

A density plot for reviews by room type shows the density of reviews for each room type. It provides a visual representation of the distribution of the reviews for each room type. This plot can help identify if there are differences in the distribution of reviews across different room types.

Side-by-side bar plots for number of bedrooms vs reviews show the distribution of the number of reviews for each number of bedrooms. It provides a visual comparison of the number of reviews for each number of bedrooms. This plot can help identify if there is a relationship betIen the number of bedrooms and the number of reviews.

Finally, a pie chart of the proportion of superhosts by room type shows the proportion of superhosts for each room type. It provides a visual summary of the distribution of superhosts across different room types. This plot can help identify if certain room types have a higher proportion of superhosts.

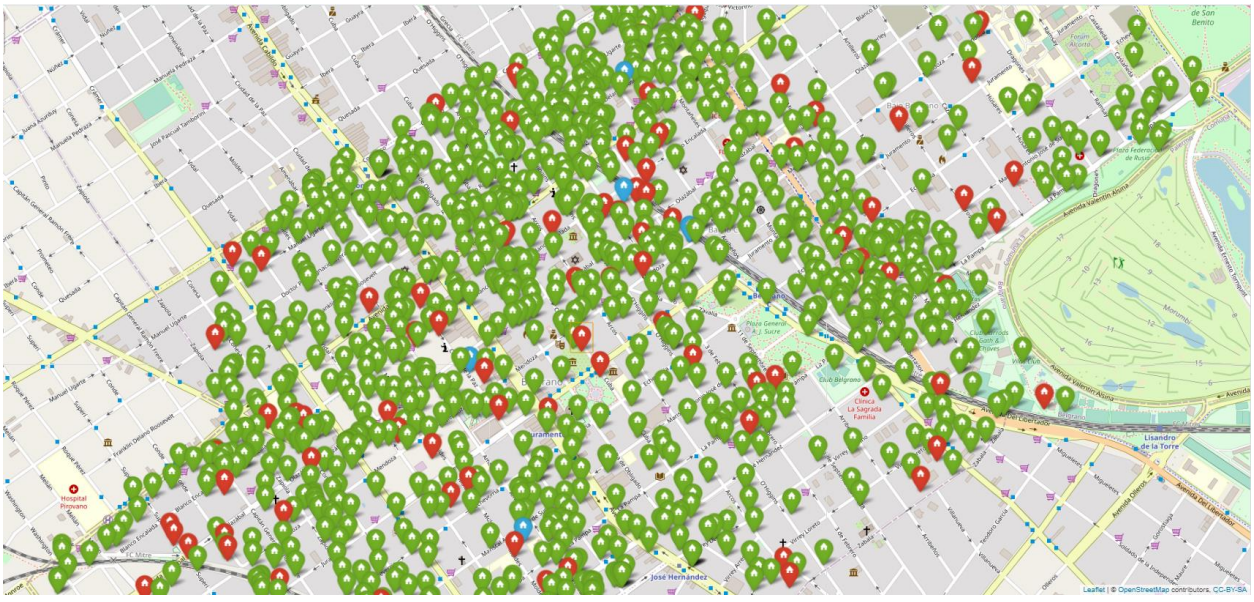
All these plots are focused on room types because they are exploring the relationship betIen different variables and the room types in a given dataset. By examining how these variables are distributed or related to each other within each room type, I can identify if there are any patterns or relationships that are specific to certain types of accommodations. For example, I may find

that certain room types have more superhosts, or that the distribution of reviews is different across different room types. This information can be useful for understanding the characteristics of different types of accommodations and how they are perceived by guests.

IV. Mapping

A. Generate a map of your neighborhood using any R mapping tool. Do any key features here seem to stand out? What are a few of the things your map shows you about the neighborhood?

```
#Part 4 Mapping

# Create map centered on Belgrano
map <- leaflet() %>%
  addTiles() %>%
  setView(-34.561042, -58.462573, zoom = 14)


# Define colors for different room types, belgrano has only three types of room
colors <- c("green", "red", "blue")

# Add markers for each listing in Belgrano
belgrano_listings <- my_data[my_data$neighbourhood_cleansed == "Belgrano",]

for (i in 1:nrow(belgrano_listings)) {
  color <- colors[match(belgrano_listings$room_type[i], c("Entire home/apt", "Private room", "Shared room"))]
  icon <- makeAwesomeIcon(icon = "home", markerColor = color, iconColor = "white", library = "ion")
  map <- addAwesomeMarkers(map, lng = belgrano_listings$longitude[i], lat = belgrano_listings$latitude[i], icon = icon)
}

# apply function to count unique values for each column
sapply(belgrano_listings, function(x) length(unique(x)))

# Display map
map
```



V. Word cloud

A. Using the neighborhood overview column in your dataset, generate a word cloud. What are some terms that seem to be emphasized here?

```
# Filter out missing values in neighborhood overview column
neighborhood_overview <- my_data %>%
  select(neighborhood_overview) %>%
  filter(!is.na(neighborhood_overview))

# Generate a word frequency table after removing stop words and stemming
word_freq <- neighborhood_overview %>%
  unnest_tokens(word, neighborhood_overview, token = "words", drop = FALSE) %>%
  filter(!word %in% c(stop_words, stopwords::stopwords("es"), stopwords::stopwords("en"))) %>%
  mutate(word = wordStem(word)) %>%
  count(word)

# Generate the wordcloud
wordcloud2(data = word_freq, size = 1.2, color = "random-dark", backgroundColor = "#f7f7f7")

# Sort word_freq by decreasing frequency
word_freq_sorted <- word_freq %>% arrange(desc(n))

# Display top 20 rows
head(word_freq_sorted, 20)
```

Based on the word cloud generated, it seems that some of the most emphasized terms are "br", "barrio", "belgrano", "cuadra", "air", "bueno", "chino", "restaurant", "ciudad", "cabildo", "zona", "palermo", "av", "avenida", "barranca", "neighborhood", "plaza", "cerca", "metro", and "river".

From this, I can draw some conclusions about the neighborhood overview descriptions of the Airbnb listings in Belgrano. The terms "barrio" and "belgrano" are used frequently, indicating that many listings may be in this specific area. The words "cuadra" (meaning block), "av" and "avenida" (meaning avenue), and "cabildo" (the name of a major street in Belgrano) suggest that the listings are located near major streets and thoroughfares. The word "barranca" likely refers to Barrancas de Belgrano, a public park in the neighborhood, and "plaza" may refer to various public squares in the area. The presence of words like "restaurant", "chino" (a reference to Chinese food or markets), and "metro" suggest that the neighborhood has a diverse range of dining options and access to public transportation. Finally, the frequent use of the term "air" may suggest that many listings in the area are promoting air conditioning as a feature.

Step II: Prediction

Initially, I have created the dataset that best suits for building the multiple linear regression with the target variable of price as taking the whole set of my_data would be headache as I have to deal with missingness and large dataset therefore, I have carried our dataset and then converted the property_type into factor and log the outcome variable as I think that it can help to transform them into a more linear relationship and I can also I can reduce the influence of extreme values and make the data more normally distributed.

```
view(my_data)
pred <- as.data.frame(my_data[,c(31,32,24,33,35,38,39,41,43,44,45,46,52,53,54,55,62,71,72)])
pred$price <- log(pred$price)
view(pred)
```

Then I have split the data into 60% associated to training set and rest of the 405 to validation set before taking the seed value as per our course as follows:

```
pred$property_type <- as.factor(pred$property_type)

set.seed(699)

train <- sample(row.names(pred), 0.6*dim(pred)[1])
valid <- setdiff(row.names(pred), train)
trainpred <- pred[train, ]
validpred <- pred[valid, ]
```

MLR Model

```
multi <- lm( price ~ . , data=trainpred)
multi
summary(multi)
```

```
Call:
lm(formula = price ~ ., data = trainpred)

Coefficients:
                                  (Intercept)                                      latitude
                                    6.299e+02                                     -3.004e+00
                                    longitude                          host_total_listings_count
                                    1.240e+01                                     -1.496e-04
                       property_typeEntire chalet                       property_typeEntire condo
                                    5.542e-01                                     -6.056e-01
                property_typeEntire guest suite                        property_typeEntire home
                                    1.268e+00                                     -1.998e-01
                       property_typeEntire loft                  property_typeEntire rental unit
                                   -4.713e-01                                     -6.064e-01
          property_typeEntire serviced apartment                     property_typeEntire townhouse
                                   -5.127e-01                                     -7.365e-01
               property_typeEntire vacation home                          property_typeEntire villa
                                   -6.345e-01                                      3.630e-01
               property_typePrivate room in barn       property_typePrivate room in casa particular
                                   -1.161e+00                                     -1.073e+00
              property_typePrivate room in condo                 property_typePrivate room in home
                                   -4.993e-01                                     -1.292e+00
        property_typePrivate room in rental unit  property_typePrivate room in serviced apartment
                                   -1.231e+00                                     -8.993e-01
          property_typePrivate room in townhouse                property_typeShared room in condo
                                   -2.934e-01                                     -1.520e+00
               property_typeShared room in home           property_typeShared room in rental unit
                                   -1.237e+00                                     -1.449e+00
                                 accommodates                                       bedrooms
                                    1.285e-01                                      2.719e-01
                                         beds                                  maximum_nights

                                         beds                                  maximum_nights
                                   -2.752e-02                                      3.073e-05
                       minimum_minimum_nights                        maximum_minimum_nights
                                    1.727e-03                                     -1.468e-03
                       minimum_maximum_nights                               availability_30
                                   -5.710e-05                                      1.331e-02
                              availability_60                               availability_90
                                   -9.949e-03                                      6.168e-03
                             availability_365                             review_scores_rating
                                    6.101e-05                                      4.466e-02
                calculated_host_listings_count  calculated_host_listings_count_entire_homes
                                    9.394e-03                                     -7.898e-03
```

Backward Elimination

```
# Backward Elimination

multi1 <- step(multi, direction="backward")
summary(multi1)
```

```
> multi1 <- step(multi, direction="backward")
Start:  AIC=-1090.42
price ~ latitude + longitude + host_total_listings_count + property_type +
    accommodates + bedrooms + beds + maximum_nights + minimum_minimum_nights +
    maximum_minimum_nights + minimum_maximum_nights + availability_30 +
    availability_60 + availability_90 + availability_365 + review_scores_rating +
    calculated_host_listings_count + calculated_host_listings_count_entire_homes

                                               Df Sum of Sq    RSS     AIC
- host_total_listings_count                     1    0.0035 130.25 -1092.4
- maximum_minimum_nights                        1    0.0149 130.26 -1092.3
- minimum_minimum_nights                        1    0.0206 130.27 -1092.3
- availability_365                              1    0.0277 130.28 -1092.3
- calculated_host_listings_count_entire_homes   1    0.0568 130.31 -1092.1
- maximum_nights                                1    0.0696 130.32 -1092.0
- calculated_host_listings_count                1    0.0814 130.33 -1092.0
- latitude                                      1    0.1190 130.37 -1091.8
- beds                                          1    0.2313 130.48 -1091.2
- minimum_maximum_nights                        1    0.2631 130.51 -1091.0
- review_scores_rating                          1    0.3332 130.58 -1090.6
<none>                                                      130.25 -1090.4
- availability_60                               1    1.0564 131.31 -1086.8
- availability_90                               1    1.4466 131.70 -1084.7
- availability_30                               1    1.9616 132.21 -1082.0
- longitude                                     1    4.6174 134.87 -1068.2
- accommodates                                  1    5.2239 135.47 -1065.0
- bedrooms                                      1    9.2969 139.55 -1044.4
- property_type                                20   25.0832 155.33 -1007.8

Step:  AIC=-1092.4
price ~ latitude + longitude + property_type + accommodates +
    bedrooms + beds + maximum_nights + minimum_minimum_nights +
    maximum_minimum_nights + minimum_maximum_nights + availability_30 +
    availability_60 + availability_90 + availability_365 + review_scores_rating +
    calculated_host_listings_count + calculated_host_listings_count_entire_homes
```

MLR with Significant Variables

```
multir <- lm(formula=price ~ longitude + property_type + accommodates + bedrooms + availability_30 +
               availability_60 + availability_90, data = trainpred)
multir
summary(multir)
```

```
Call:
lm(formula = price ~ longitude + property_type + accommodates +
    bedrooms + availability_30 + availability_60 + availability_90,
    data = trainpred)

Coefficients:
                                  (Intercept)                                     longitude
                                   684.008640                                    11.549641
                  property_typeEntire chalet                  property_typeEntire condo
                                     0.635672                                    -0.572382
             property_typeEntire guest suite                   property_typeEntire home
                                     1.320945                                    -0.158309
                    property_typeEntire loft              property_typeEntire rental unit
                                    -0.428331                                    -0.565982
      property_typeEntire serviced apartment                property_typeEntire townhouse
                                    -0.490751                                    -0.749225
           property_typeEntire vacation home                    property_typeEntire villa
                                    -0.567330                                     0.479067
           property_typePrivate room in barn  property_typePrivate room in casa particular
                                    -1.139970                                    -1.013443
          property_typePrivate room in condo            property_typePrivate room in home
                                    -0.495672                                    -1.229586
    property_typePrivate room in rental unit property_typePrivate room in serviced apartment
                                    -1.201940                                    -0.891441
      property_typePrivate room in townhouse            property_typeShared room in condo
                                    -0.292214                                    -1.534609
           property_typeShared room in home      property_typeShared room in rental unit
                                    -1.178766                                    -1.476892
                                 accommodates                                     bedrooms
                                     0.111565                                     0.262251
                              availability_30                                availability_60
                                     0.013474                                    -0.009720
                              availability_90
```

MLR without property_type

```
# MLR Model without property_type

multip <- lm(formula=price ~ accommodates + bedrooms + availability_30 +
               availability_60 + availability_90, data=trainpred)
multip
summary(multip)
```

```
> multip

Call:
lm(formula = price ~ accommodates + bedrooms + availability_30 +
    availability_60 + availability_90, data = trainpred)

Coefficients:
    (Intercept)     accommodates         bedrooms  availability_30  availability_60  availability_90
       8.110745         0.168721         0.261167         0.013192        -0.010015         0.006436
```

```
> summary(multip)

Call:
lm(formula = price ~ accommodates + bedrooms + availability_30 +
    availability_60 + availability_90, data = trainpred)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0949 -0.2451 -0.0262  0.2617  3.1670

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     8.110745   0.056399 143.809  < 2e-16 ***
accommodates    0.168721   0.019838   8.505  < 2e-16 ***
bedrooms        0.261167   0.038121   6.851 1.63e-11 ***
availability_30 0.013192   0.004545   2.902  0.00382 **
availability_60 -0.010015  0.004633  -2.162  0.03097 *
availability_90 0.006436   0.002360   2.727  0.00655 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4907 on 690 degrees of freedom
Multiple R-squared:  0.4048,    Adjusted R-squared:  0.4005
F-statistic: 93.85 on 5 and 690 DF,  p-value: < 2.2e-16
```

Accuracy of our model against both the training set and the validation set.

```
> predac<-predict(multip, trainpred)
> accuracy(predac ,trainpred$price)
                      ME      RMSE       MAE         MPE      MAPE
Test set -1.074495e-14 0.488574 0.3519452 -0.2867679 3.862171
> predac1<-predict(multip, validpred)
> accuracy(predac1 ,validpred$price)
                    ME      RMSE       MAE       MPE      MAPE
Test set 0.05988757 0.5286247 0.3879386 0.3331365 4.224964
```

The various model I have build are based on different ideas and perspectives as the first model shows that, I have taken all the variables against the target variable in which our model was able

to capture the 0.4449 as residual standard error, through which I can understand that our model predicted that all the variables I have used are the average deviation of approximately 0.49 units from the price of the rentals in Belgrano which is pretty good as the value is not much big means further means that a better fit to our model.

Secondly, I have carried out the backward elimination model as I wanted that the model should remove all the predictors variables from the model that do not contribute significantly to the model performance, so that I can make the upcoming model with valid and appropriate accuracy level and on other I are also saved or minimized the situation of overfitting in the model, as I think that model performs Ill on the training data but poorly on validation data, and I have also used backward elimination in our previous assignment.

Additionally, I have created the new model based on several variables that I think would I better align to the future results and simultaneously, they Ire also having significant impact on the model as their p-value was less than 5%, therefore, I have selected (longitude, property_type, accommodates, availability_30, availability_60 and availability_90 as I get the suitable result from the backward elimination model and this model carries the quality residual standard error of o.4907 and very low p-value, and the multiple R-square was of 0.5273, which means that approximately 52% of the variability of price can be highlighted and explained by all the variables I have used in the model.

Finally, I have built our final model, but this I haven't included the property_type and longitude as Ill because I think that the these variable have so many elements which can further invite complications in the model, which our team doesn't like and it can also reduce the statistical poIr of the model as the property_type can increase the noise in the model and reduce the signal, and making us in more difficult shape, as it is hard for us to detect significant effects.

This model has the multiple R squared of 0.4048, through which I can consider that there is around good chances that variables have control over the price of rentals and it can influence as per the level of accommodates and availability specially in Belgrano and on other hand I look towards accuracy of the model the ME highlight really small value which mean that our model is predicting the values accurately in terms of average difference betIen our values and the predictor value I have taken in this model and RMSE value reflects the average squared difference betIen the values and the predicted values I have taken in the model and taking the square root of the result , thus I gently believe that RMSE value in this case is 0.488574, which suggests that the model's predictions are, on right tracks.

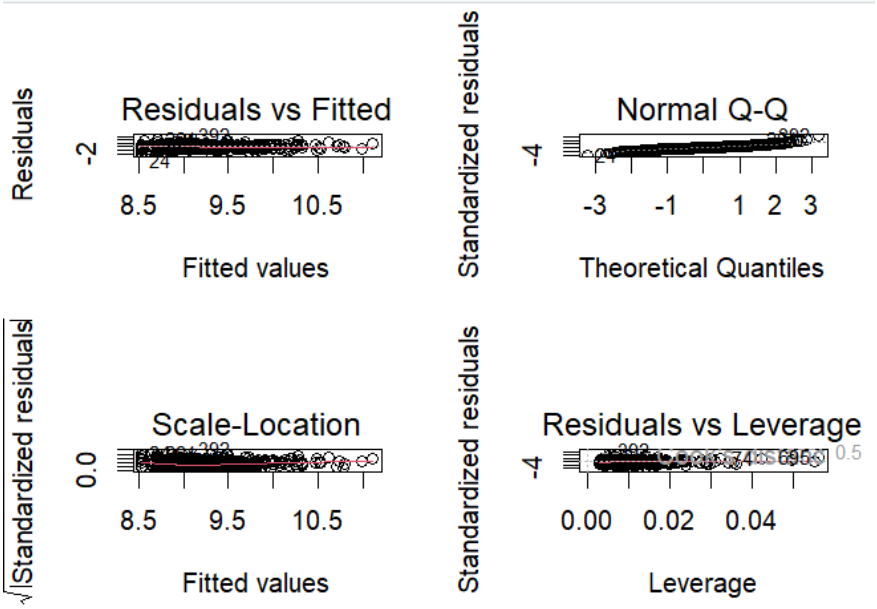
Moreover, MAE that I know measures the average absolute difference and in this case our MAE is 0.3519452, which suggests that the model's predictions are, on good level and then the MPE value in this case is -0.2867679, which suggests that the model is underestimating the values by 0.2867679% and then finally MAPE in this case is 3.862171, which suggests that the model's predictions are within 3.862171% of the actual values that I have taken.

Calculation of SST, SSR and SSE

```
> SST1<-sum((trainpred$price-mean(trainpred$price))^2)
> print(SST1)
[1] 279.1199
> SSR1<-sum((multip$fitted.values-mean(trainpred$price))^2)
> print(SSR1)
[1] 112.9816
> SSE<-sum((multip$residuals)^2)
> print(SSE)
[1] 166.1383
> SSR1/SST1
[1] 0.4047779
```

I have calculated the SST (Total Sum of Squares) that is total variation in the dependent variable I have used and also calculated the SSR (Regression Sum of Squares) which is SSR is calculated as the sum of the squared differences betIen the predicted values I have taken and the designated mean of them and when I divide both the value I get 0.4047, through which I can understand that our model has explained 40% of the variation in the set of variable I have taken, as calculated by the R-squared value.

Our Model Plot



Step III: Classification

1 Firstly I have created the datasets that contains the valid and suitable variable through which the model can generate quality output that can significantly impact the efficient and performance of the knn model.

```
knear <- as.data.frame(my_data[,c(24,35,38,39,40,41,45,54,57,64,72)])
view(knear)
```

Then I have split the dataset with assigning 60% to training set and remaining 40% to validation set after setting the seed value.

```
set.seed(699)

# Splitting the data

train <- sample(row.names(knear), 0.6*dim(knear)[1])
valid <- setdiff(row.names(knear), train)
train <- knear[train, ]
valid <- knear[valid, ]
```

Secondly, I have created the set of amenities in combination set to TV and Wifi that reflects that whether a rental in Belgrano will have some particular amenity, or combination of amenities and it was possible with the help of grepl function as I mutate the set of amenities as 1 means there and 0 means not there in the dataset.

```
train1 <- train %>% mutate(tv_there= ifelse(grepl("TV|Wifi",amenities), '1', '0'))
valid1 <- valid %>% mutate(tv_there= ifelse(grepl("TV|Wifi",amenities), '1', '0'))
```

```
# placing as factor

class(train1$tv_there)
train1$tv_there <- as.factor(train1$tv_there)
valid1$tv_there <- as.factor(valid1$tv_there)
```

Then I have checked the mean impact in the dataset by identifying the significant impact of each of the variable as I think that variables with high p-values are not statistically significant and are unlikely to have a strong impact on the target variable therefore, removing these variables can simplify the model and reduce the risk of overfitting.

```
good <- filter(train1, tv_there == "1")
no_good <- filter(train1, tv_there == "0")
view(train1)


num_vars <- c("host_total_listings_count" ,
              "accommodates", "bedrooms", "beds", "price",
              "maximum_minimum_nights", "availability_90", "number_of_reviews",
              "review_scores_cleanliness",
              "calculated_host_listings_count_entire_homes")

for (var in num_vars) {
  t_test <- t.test(good[[var]], no_good[[var]])
  cat(var, "\n")
  cat("p-value:", t_test$p.value, "\n\n")
}
```

```
host_total_listings_count
p-value: 7.868784e-14

accommodates
p-value: 0.0005698277

bedrooms
p-value: 0.3462485

beds
p-value: 0.0007616313

price
p-value: 0.04432941

maximum_minimum_nights
p-value: 0.5477572

availability_90
p-value: 0.07410524

number_of_reviews
p-value: 3.697751e-10

review_scores_cleanliness
p-value: 0.6589024

calculated_host_listings_count_entire_homes
p-value: 2.71464e-19
```

Now I can take an informed decision to remove the (bedrooms, maximum_minimum_nights, availability_90, review_scores_cleanliness) and keep the rest of the variables as there the p-values are less than the significance level, I reject the null hypothesis and conclude that there is significant evidence to support the alternative hypothesis.

```
train1 = subset(train1, select = -c(bedrooms, maximum_minimum_nights, availability_90, review_scores_cleanliness))
valid1 = subset(valid1,select= -c(bedrooms, maximum_minimum_nights, availability_90, review_scores_cleanliness))
knear = subset(knear,select= -c(bedrooms, maximum_minimum_nights, availability_90, review_scores_cleanliness))
```

Afterwards, I have normalized the dataset including training and validation set by using preprocess function and predict the values.

```
# Normalizing all the available Dataset.

knear.new<-data.frame(host_total_listings_count=16,
                      accommodates=3,beds=1,price=4467,number_of_reviews= 20,
                      calculated_host_listings_count_entire_homes=14)

train.norm <- train1
valid.norm <- valid1
knear.norm <- knear

# Now I have undertaken preProcess() from the caret package to normalize the data as follows:

norm.values <- preProcess(train1[,c(1:3,5,6,7)], method=c("center", "scale"))
train.norm[,c(1:3,5,6,7)] <- predict(norm.values, train1[,c(1:3,5,6,7)])
valid.norm[,c(1:3,5,6,7)] <- predict(norm.values, valid1[,c(1:3,5,6,7)])
knear.norm[,c(1:3,5,6,7)] <- predict(norm.values, knear[,c(1:3,5,6,7)])
knk.norm<- predict(norm.values, knear.new)
train.norm <- as.data.frame(train.norm)
view(train1)
```

KNN Function

```
knn<- FNN::knn(train = train.norm[,c(1:3,5,6,7)], test = knk.norm,
               cl = train.norm$tv_there, k = 6, prob = TRUE)
knn
```

```
> knn
[1] 1
attr(,"prob")
[1] 1
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   49  433   42  576  102   56
attr(,"nn.dist")
     [,1]      [,2]      [,3]     [,4]      [,5]      [,6]
[1,]    0 0.4411444 0.6877775 0.690116 0.6983907 0.7030728
Levels: 1
```

Then I have presented the list of all the six nearest neighbors of the model I just created through which I redefine our understanding in terms of combination of the amenities.

```
neigh <- train.norm[c(49 , 433 ,  42,   576,  102,   56),]
neigh
```

```
    host_total_listings_count accommodates       beds
912               -0.02980935   0.03400485 -0.7888271
769                0.09547631   0.03400485 -0.7888271
983               -0.18015213   0.03400485 -0.7888271
485               -0.38060918   0.03400485 -0.7888271
713               -0.40566632   0.03400485 -0.7888271
81                -0.40566632   0.03400485 -0.7888271

amenities
912
["Extra pillows and blankets", "Dishes and silverware", "Room-darkening shades", "Hot water kettle", "First
aid kit", "Ethernet connection", "Wifi", "Oven", "Dining table", "Fire extinguisher", "Clothing storage: clo
set", "AC - split type ductless system", "Kitchen", "Cooking basics", "Pets allowed", "Coffee maker", "Smoke
alarm", "Luggage dropoff allowed", "Cleaning products", "Heating", "Shampoo", "Shower gel", "Baking sheet",
"Essentials", "Hair dryer", "Bed linens", "City skyline view", "Body soap", "Paid parking lot off premises",
"Self check-in", "Microwave", "Lockbox", "Hangers", "Electric stove", "Refrigerator", "Wine glasses", "Long
term stays allowed", "Drying rack for clothing", "Conditioner", "Hot water", "TV with standard cable"]
769
["Extra pillows and blankets", "Dishes and silverware", "Room-darkening shades", "Hot water kettle", "Wifi",
"Dedicated workspace", "Paid washer \\u2013 In building", "Outdoor furniture", "Fire extinguisher", "Mini fr
idge", "Elevator", "Free street parking", "AC - split type ductless system", "Kitchen", "Cooking basics", "P
ets allowed", "Paid parking off premises", "Toaster", "Radiant heating", "Cleaning products", "Shampoo", "Sh
ower gel", "Clothing storage", "43\\" HDTV with Netflix, standard cable", "Essentials", "Hair dryer", "Paid
dryer \\u2013 In building", "Bed linens", "Paid parking on premises", "Body soap", "Private patio or balcon
y", "Bidet", "Self check-in", "Microwave", "Coffee maker: Nespresso", "Hangers", "Electric stove", "Keypad",
"Laundromat nearby", "Refrigerator", "Wine glasses", "Long term stays allowed", "Coffee", "Conditioner", "Ho
t water"]

983 ["Coffee maker: drip coffee maker", "Extra pillows and blankets", "Dishes and silverware", "Room-darkeni
ng shades", "Hot water kettle", "First aid kit", "Wifi", "Dedicated workspace", "Paid washer \\u2013 In buil
ding", "Heating - split type ductless system", "Fire extinguisher", "Freezer", "Elevator", "Free street park
ing", "Cleaning available during stay", "Kitchen", "Building staff", "Cooking basics", "Toaster", "Exercise
equipment", "Smoke alarm", "Bathtub", "Crib", "Cleaning products", "Stainless steel oven", "Shampoo", "Share
d gym in building", "Luggage dropoff allowed", "Shower gel", "TV", "Baking sheet", "Essentials", "Hair drye
r", "Paid dryer \\u2013 In building", "Bed linens", "Carbon monoxide alarm", "Body soap", "Private patio or
balcony", "Bidet", "Self check-in", "Microwave", "Hangers", "Electric stove", "Iron", "Air conditioning", "R
efrigerator", "Blender", "Wine glasses", "Long term stays allowed", "Drying rack for clothing", "Coffee", "C
onditioner", "Clothing storage: closet and dresser", "Hot water"]
485
["Extra pillows and blankets", "Dishes and silverware", "Hot water kettle", "Wifi", "Crib - available upon r
equest", "Dedicated workspace", "Dining table", "Shared pool", "Paid dryer", "Freezer", "Elevator", "Free st
reet parking", "Kitchen", "Paid parking lot on premises", "Cooking basics", "Single level home", "Pets allow
ed", "Security cameras on property", "Toaster", "Bathtub", "Central air conditioning", "Cleaning products",
"Heating", "Stainless steel oven", "Shower gel", "TV", "Essentials", "Hair dryer", "Bed linens", "BBQ gril
l", "Body soap", "Private patio or balcony", "Bidet", "Pack \\u2019n play/Travel crib", "Microwave", "Hanger
s", "Iron", "Courtyard view", "Refrigerator", "Wine glasses", "Paid washer", "Long term stays allowed", "Dry
ing rack for clothing", "Baking sheet", "Hot water", "Clothing storage: closet"]
713
["Dishes and silverware", "Hot water kettle", "First aid kit", "Ethernet connection", "Wifi", "Dedicated wor
kspace", "Fire extinguisher", "Freezer", "Host greets you", "Kitchen", "Cooking basics", "Pool", "Coffee mak
er", "Toaster", "Smoke alarm", "TV", "Essentials", "Hair dryer", "Bed linens", "Carbon monoxide alarm", "Bid
et", "Hangers", "Air conditioning", "Refrigerator", "Outdoor dining area", "Wine glasses", "Heating", "Hot w
ater", "Clothing storage: closet"]
81
["Dishes and silverware", "Wifi", "Oven", "Dedicated workspace", "Free street parking", "Stove", "Kitchen",
"Cooking basics", "Pets allowed", "Paid parking off premises", "Private hot tub", "Coffee maker", "Shampoo",
"TV", "Essentials", "Hair dryer", "Children\\u2019s books and toys", "City skyline view", "Paid parking on p
remises", "Private patio or balcony", "Hangers", "Iron", "Air conditioning", "Refrigerator", "Outdoor dining
area", "Backyard", "Long term stays allowed", "Heating", "Hot water", "Indoor fireplace"]
        price number_of_reviews calculated_host_listings_count_entire_homes tv_there
```

	price	number_of_reviews	calculated_host_listings_count_entire_homes	tv_there
912	-0.5170699	0.155877279	0.1452729	1
769	-0.2168922	0.002750128	0.4009238	1
983	-0.2228330	-0.341785959	-0.1955950	1
485	-0.3195547	-0.073813447	-0.3660289	1
713	-0.3216240	0.194159066	-0.4086374	1
81	-0.6149930	-0.035531659	-0.4086374	1

I have determined whether the knn model have the combination set of amenities of TRv and Wifi therefore, yes, they have them.

```
> if (knn == 1) {
+   print("great")
+ } else {
+   print("not great")
+ }
[1] "great"
```

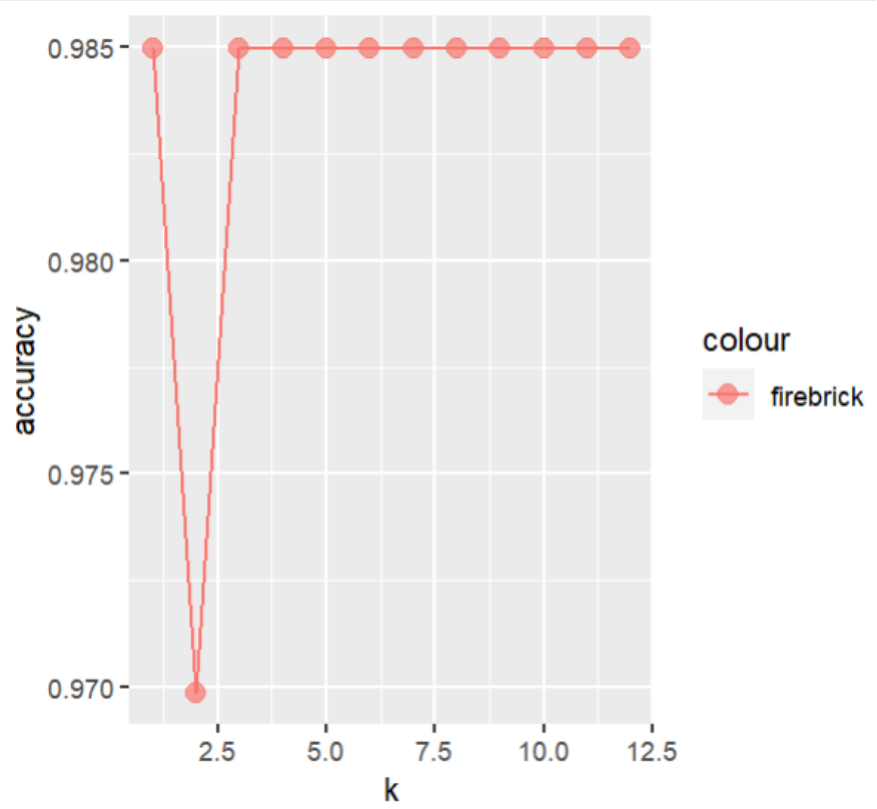
Finally, I have defined the accuracy of the model to evaluate the performance of a k-nearest neighbors of the set of amenities.

```
accuracy <- data.frame(k = seq(1, 12, 1), accuracy = rep(0, 12))

for(i in 1:12) {
  knear.pred <- FNN::knn(train.norm[,c(1:3,5,6,7)], valid.norm[,c(1:3,5,6,7)],
                         cl = train.norm$tv_there, k = i)
  accuracy[i, 2] <- confusionMatrix(knear.pred, valid.norm$tv_there)$overall[1]
}

print(accuracy)
```

```
> print(accuracy)
    k  accuracy
1   1 0.9849462
2   2 0.9698925
3   3 0.9849462
4   4 0.9849462
5   5 0.9849462
6   6 0.9849462
7   7 0.9849462
8   8 0.9849462
9   9 0.9849462
10 10 0.9849462
11 11 0.9849462
12 12 0.9849462
```

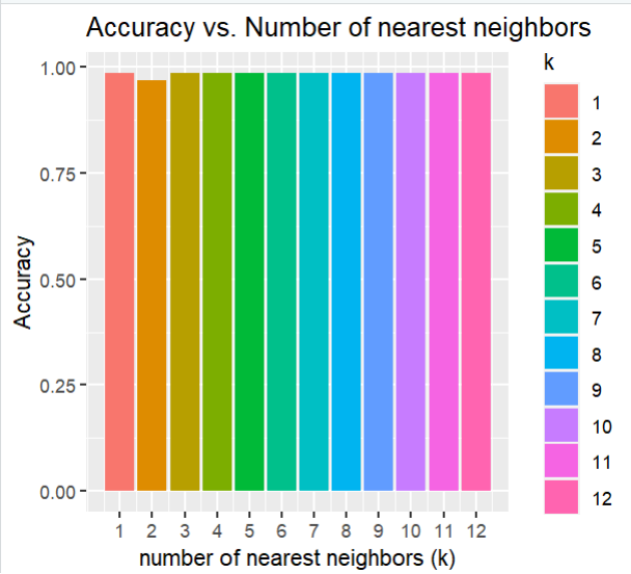
Here are some crucial visualizations that are worth of thousand words.

```
ggplot(accuracy,  aes(x=k, y=accuracy, color = "firebrick")) +
  geom_point(size = 3, alpha = 0.7) + geom_line()
```



Second Visualization

```
ggplot(accuracy, aes(x = k, y = accuracy, fill = factor(k))) +
  geom_bar(stat = "identity") +
  scale_x_continuous(breaks = seq(1, 12, 1)) +
  xlab("number of nearest neighbors (k)") +
  ylab("Accuracy") +
  ggtitle("Accuracy vs. Number of nearest neighbors") +
  scale_fill_discrete(name = "k")
```



HoIver, I can interpret that K-3 is the most solid among other so, I are moving forward with it.

```
knn2<- FNN::knn(train = train.norm[,c(1:3,5,6,7)], test = knk.norm,
                cl = train.norm$tv_there, k = 3)
knn2
```

```
[1] 1
attr(,"nn.index")
     [,1] [,2] [,3]
[1,]   49  433   42
attr(,"nn.dist")
     [,1]      [,2]      [,3]
[1,]    0 0.4411444 0.6877775
Levels: 1
```

Again, the nearest neighbors are as follows:

```
neigh1 <- train.norm[c(49 , 433 ,  42),]
neigh1
```

```
    host_total_listings_count accommodates       beds
912               -0.02980935   0.03400485 -0.7888271
769                0.09547631   0.03400485 -0.7888271
983               -0.18015213   0.03400485 -0.7888271
```

```
amenities
912
["Extra pillows and blankets", "Dishes and silverware", "Room-darkening shades", "Hot water kettle",
"First aid kit", "Ethernet connection", "Wifi", "Oven", "Dining table", "Fire extinguisher", "Clothing
storage: closet", "AC - split type ductless system", "Kitchen", "Cooking basics", "Pets allowed", "Cof
fee maker", "Smoke alarm", "Luggage dropoff allowed", "Cleaning products", "Heating", "Shampoo", "Show
er gel", "Baking sheet", "Essentials", "Hair dryer", "Bed linens", "City skyline view", "Body soap",
"Paid parking lot off premises", "Self check-in", "Microwave", "Lockbox", "Hangers", "Electric stove",
"Refrigerator", "Wine glasses", "Long term stays allowed", "Drying rack for clothing", "Conditioner",
"Hot water", "TV with standard cable"]
769
["Extra pillows and blankets", "Dishes and silverware", "Room-darkening shades", "Hot water kettle",
"Wifi", "Dedicated workspace", "Paid washer \\u2013 In building", "Outdoor furniture", "Fire extinguis
her", "Mini fridge", "Elevator", "Free street parking", "AC - split type ductless system", "Kitchen",
"Cooking basics", "Pets allowed", "Paid parking off premises", "Toaster", "Radiant heating", "Cleaning
products", "Shampoo", "Shower gel", "Clothing storage", "43\\" HDTV with Netflix, standard cable", "Es
sentials", "Hair dryer", "Paid dryer \\u2013 In building", "Bed linens", "Paid parking on premises",
"Body soap", "Private patio or balcony", "Bidet", "Self check-in", "Microwave", "Coffee maker: Nespres
so", "Hangers", "Electric stove", "Keypad", "Laundromat nearby", "Refrigerator", "Wine glasses", "Long
term stays allowed", "Coffee", "Conditioner", "Hot water"]
983 ["Coffee maker: drip coffee maker", "Extra pillows and blankets", "Dishes and silverware", "Room-d
arkening shades", "Hot water kettle", "First aid kit", "Wifi", "Dedicated workspace", "Paid washer \\u
2013 In building", "Heating - split type ductless system", "Fire extinguisher", "Freezer", "Elevator",
"Free street parking", "Cleaning available during stay", "Kitchen", "Building staff", "Cooking basic
s", "Toaster", "Exercise equipment", "Smoke alarm", "Bathtub", "Crib", "Cleaning products", "Stainless
steel oven", "Shampoo", "Shared gym in building", "Luggage dropoff allowed", "Shower gel", "TV", "Baki
ng sheet", "Essentials", "Hair dryer", "Paid dryer \\u2013 In building", "Bed linens", "Carbon monoxid
e alarm", "Body soap", "Private patio or balcony", "Bidet", "Self check-in", "Microwave", "Hangers",
"Electric stove", "Iron", "Air conditioning", "Refrigerator", "Blender", "Wine glasses", "Long term st
ays allowed", "Drying rack for clothing", "Coffee", "Conditioner", "Clothing storage: closet and dress
er", "Hot water"]
```

	price	number_of_reviews	calculated_host_listings_count_entire_homes	tv_there
912	-0.5170699	0.155877279	0.1452729	1
769	-0.2168922	0.002750128	0.4009238	1
983	-0.2228330	-0.341785959	-0.1955950	1

In the above code, I have used k-nearest neighbors (k-NN) algorithm to predict whether a rental in Belgrano will have a particular amenity or not as I have selected the numerical predictors such as host_total_listing_count, accommodated, beds, price, number_of_reviews and calculated_host_listing_count_entire_homesto build the model and the outcome variable is the amenities column, I have then used the "grepl" function to create a binary indicator variable for set and TV and Wifi set and I decided to move forward with 3 k as I believe they have significant impact in the model as I have interpret form both the visualization of the model in which 3 k is most solid one and then I perform the accuracy level.

Moreover, the the values of k and the corresponding accuracy of the model for each value of k the accuracy values are ranging from 0.969 to 0.985, indicating that the model is performing very Ill in predicting the advantage that most of the rentals in Belgrano have Tv and Wifi which are the most necessary facilities in the present generation which also means the rental properties have to maintain quality amount of expenses in terms of maintenance, electricity and subscription charges which connects the price factor of the accommodation but on other hand the competition thrives and enables to adapt the changes with minimum rates but tend to limited usage as the analysis form the model highlights that only nine rentals does the provide the set amenities of Tv and Wifi and means very few of them are might running out of the fashion or have limited resources or they wany simple life with peace without any modern amenities to them and to guest as Ill hoIver, this model can provide lots of significant information that an hotel organization can take and alter their strategies and make significant decisions.

Classification, Part II. Naive Bayes

Firstly, I have created the datasets that contains the valid and suitable variable through which the model can generate quality output that can significantly impact the efficient and performance of the naïve model in which I have undertaken the factors variable as naïve bayes model performs only and Ill with categorical values.

```
naive <- as.data.frame(my_data[,c(33, 34, 35, 37, 57,70)])

naive$property_type <- as.factor(naive$property_type)
naive$room_type <- as.factor(naive$room_type)
naive$accommodates <- as.factor(naive$accommodates)
naive$bathrooms_text <- as.factor(naive$bathrooms_text)
naive$number_of_reviews <- as.factor(naive$number_of_reviews)
naive$instant_bookable <- as.factor(naive$instant_bookable)
```

Then I have split the dataset into training set with 60% and rest of the 40% to validation set.

```
train <- sample(row.names(naive), 0.6*dim(naive)[1])
valid <- setdiff(row.names(naive), train)
trainai <- naive[train, ]
validnai <- naive[valid, ]
```

Naïve Bayes Model

```
nai_b <- naiveBayes(instant_bookable ~ ., data= trainai)
nai_b
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    FALSE      TRUE
0.7212644 0.2787356

Conditional probabilities:
       property_type
Y       Casa particular Entire chalet Entire condo Entire guest suite Entire home Entire loft
  FALSE     0.001992032   0.000000000  0.181274900        0.001992032 0.029880478 0.031872510
  TRUE      0.000000000   0.000000000  0.128865979        0.000000000 0.010309278 0.020618557
       property_type
Y       Entire place Entire rental unit Entire serviced apartment Entire townhouse
  FALSE  0.000000000        0.585657371              0.047808765      0.001992032
  TRUE   0.000000000        0.670103093              0.036082474      0.005154639
       property_type
Y       Entire vacation home Entire villa Private room in barn Private room in casa particular
  FALSE          0.011952191  0.003984064          0.000000000                     0.015936255
  TRUE           0.072164948  0.000000000          0.005154639                     0.005154639
       property_type
Y       Private room in condo Private room in home Private room in loft
  FALSE           0.009960159          0.009960159          0.000000000
  TRUE            0.010309278          0.010309278          0.000000000
       property_type
Y       Private room in rental unit Private room in serviced apartment Private room in townhouse
  FALSE                0.053784861                       0.001992032              0.001992032
  TRUE                 0.005154639                       0.010309278              0.000000000
```

Furthermore, I have carried out the confusion matrix that will compares the performance of our naïve bayes model against the training data, and another that shows its performance against the validation data.

```
trainpd <- predict(nai_b, newdata= trainai)
confusionMatrix(trainpd, trainai$instant_bookable)
```

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   486  165
     TRUE     16   29

               Accuracy : 0.7399
                 95% CI : (0.7057, 0.7722)
    No Information Rate : 0.7213
    P-Value [Acc > NIR] : 0.1451

                  Kappa : 0.1539

 Mcnemar's Test P-Value : <0.0000000000000002

            Sensitivity : 0.9681
            Specificity : 0.1495
         Pos Pred Value : 0.7465
         Neg Pred Value : 0.6444
             Prevalence : 0.7213
         Detection Rate : 0.6983
   Detection Prevalence : 0.9353
      Balanced Accuracy : 0.5588

       'Positive' Class : FALSE
```

```
# Validation set

validpd <- predict(nai_b, newdata=  validnai)
confusionMatrix(validpd,  validnai$instant_bookable)
```

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   324  111
     TRUE     22    8

               Accuracy : 0.714
                 95% CI : (0.6706, 0.7547)
    No Information Rate : 0.7441
    P-Value [Acc > NIR] : 0.937

                  Kappa : 0.0048

 Mcnemar's Test P-Value : 0.00000000000002337

            Sensitivity : 0.93642
            Specificity : 0.06723
         Pos Pred Value : 0.74483
         Neg Pred Value : 0.26667
             Prevalence : 0.74409
         Detection Rate : 0.69677
   Detection Prevalence : 0.93548
      Balanced Accuracy : 0.50182

       'Positive' Class : FALSE
```

Then I have carried out the fictional apartment of Belgrano in which I have taken points as follows:

1. Property_type: Entire Condo
2. Room_type: Entire home/apt
3. Accommodates: 4
4. Bathroom_Text: 2
5. Number_of_reviews: 42

```
fictionalap <- data.frame(property_type="Entire rental unit", room_type= "Entire home/apt",
                          accommodates= "2", bathrooms_text= "1", number_of_reviews="104")
```

```
ficpd <- predict(nai_b, newdata = fictionalap , type = "raw")
ficpd
```

```
          FALSE      TRUE
[1,] 0.6702471 0.3297529
```

In the naïve bayes model as an group I have initially cross selected the suitable variables to build the naïve bayes model and then I know that the model won't perform with the presence of numerical variables, so I have converted all the variables to factors as the algorithm calculates probabilities based on the frequency of occurrences of different categories I have taken in the training data and since probabilities can only be calculated for categorical variables, the algorithm requires the input data to be in categorical format and then I have split the data into training and validation set and then I have performed the naïve bayes model that gives us 73.99% accuracy associated with training set and when I compared it with the validation set the model provides us 71% accuracy which not pretty bad therefore, I can consider that our model is on right track thus, rentals in Belgrano can be booked immediately whenever sudden plan comes out and people can find and book rental flexibility as there are lots of opportunities in terms of business, education and tourism as it's an major landmark that attracts lots of tourists.

Furthermore, I have created the fictional apartment with suitable variables through which I can determine that what our predicts in terms of fictional data and this can be very useful to identify the chances of accuracy, success, and demand whenever company id trying to invest in major rentals accommodates or build new place for people as they have an extra edge to make certain assumptions baes on historical data.

Classification, Part III. Classification Tree

Initially, I have selected the suitable variables to build the classification tree as per the review scores rating.

```
class <- as.data.frame(my_data[,c(19,24,27,33,34,35,37,38,39,41,43,44,45,
                                  46,51,52,53,54,55,62,70,71,72)])
```

Then I have binned the frequency to the review scores rating and split the dataset.

```
class$review_scores_rating<-discretize(class$review_scores_rating, method="frequency",
                                       breaks=4, labels=c("poor","average","good"))
```

```
# Split the Dataset

set.seed(699)


train <- sample(row.names(class), 0.6*dim(class)[1])
valid <- setdiff(row.names(class), train)
traintree <- class[train, ]
validtree <- class[valid, ]
```

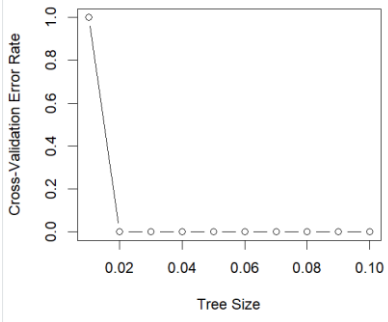
Classification Tree

```
tree<-rpart(review_scores_rating~.,method="class",data=traintree)
tree
```

```
n= 696

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 696 352 good (0.22270115 0.28304598 0.49425287)
    2) calculated_host_listings_count_entire_homes>=7.5 162 100 good (0.36419753 0.25308642 0.38271605)
      4) maximum_nights>=105 138  81 poor (0.41304348 0.20289855 0.38405797)
        8) host_is_superhost< 0.5 68  30 poor (0.55882353 0.13235294 0.30882353) *
        9) host_is_superhost>=0.5 70  38 good (0.27142857 0.27142857 0.45714286)
         18) host_total_listings_count< 26.5 37  24 poor (0.35135135 0.35135135 0.29729730)
           36) availability_30>=5.5 27  14 poor (0.48148148 0.37037037 0.14814815) *
           37) availability_30< 5.5 10   3 good (0.00000000 0.30000000 0.70000000) *
         19) host_total_listings_count>=26.5 33  12 good (0.18181818 0.18181818 0.63636364) *
      5) maximum_nights< 105 24  11 average (0.08333333 0.54166667 0.37500000)
       10) host_total_listings_count>=39.5 9   1 average (0.11111111 0.88888889 0.00000000) *
       11) host_total_listings_count< 39.5 15   6 good (0.06666667 0.33333333 0.60000000) *
    3) calculated_host_listings_count_entire_homes< 7.5 534 252 good (0.17977528 0.29213483 0.52808989)
      6) host_is_superhost< 0.5 375 174 good (0.22400000 0.24000000 0.53600000) *
      7) host_is_superhost>=0.5 159  78 good (0.07547170 0.41509434 0.50943396)
       14) maximum_nights< 105 54  23 good (0.16666667 0.25925926 0.57407407) *
       15) maximum_nights>=105 105  53 average (0.02857143 0.49523810 0.47619048)
         30) minimum_maximum_nights< 362.5 10   1 average (0.00000000 0.90000000 0.10000000) *
         31) minimum_maximum_nights>=362.5 95  46 good (0.03157895 0.45263158 0.51578947)
           62) beds< 1.5 39  15 average (0.02564103 0.61538462 0.35897436)
            124) minimum_minimum_nights< 4.5 25   5 average (0.04000000 0.80000000 0.16000000) *
            125) minimum_minimum_nights>=4.5 14   4 good (0.00000000 0.28571429 0.71428571) *
           63) beds>=1.5 56  21 good (0.03571429 0.33928571 0.62500000) *
```

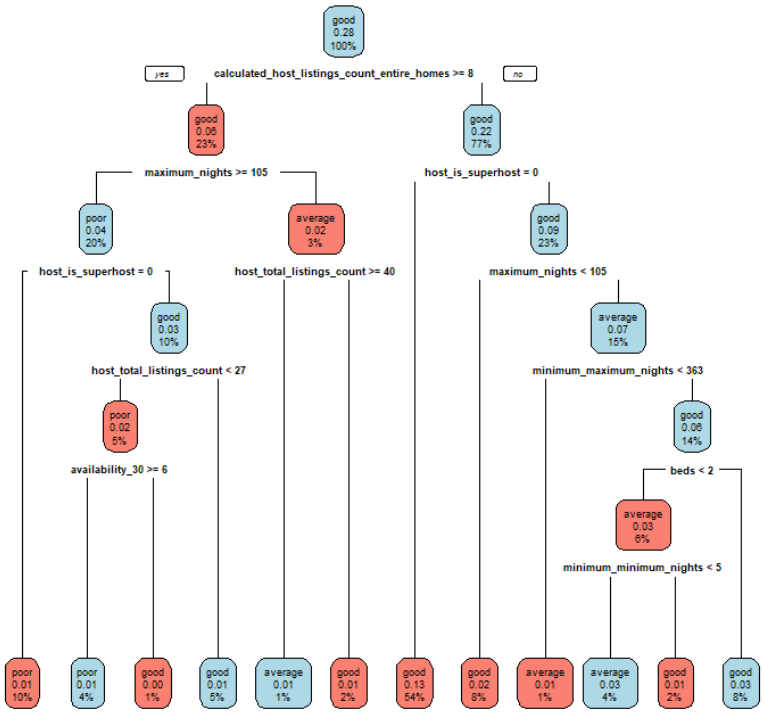
Cross-Validation

```
cv_results <- data.frame(size = integer(), error_rate = double())

for (i in 1:10) {
  set.seed(699)
  cv_model <- rpart(review_scores_rating ~ ., data = traintree, method = "class", cp = i/100)
  cv_results <- rbind(cv_results, data.frame(size = i/100, error_rate =
                                             cv_model$cptable[which.min(cv_model$cptable[,"xerror"]),"xerror"]))
}

plot(cv_results$size, cv_results$error_rate, type = "b", xlab = "Tree Size",
     ylab = "Cross-Validation Error Rate")

print(cv_results)
```



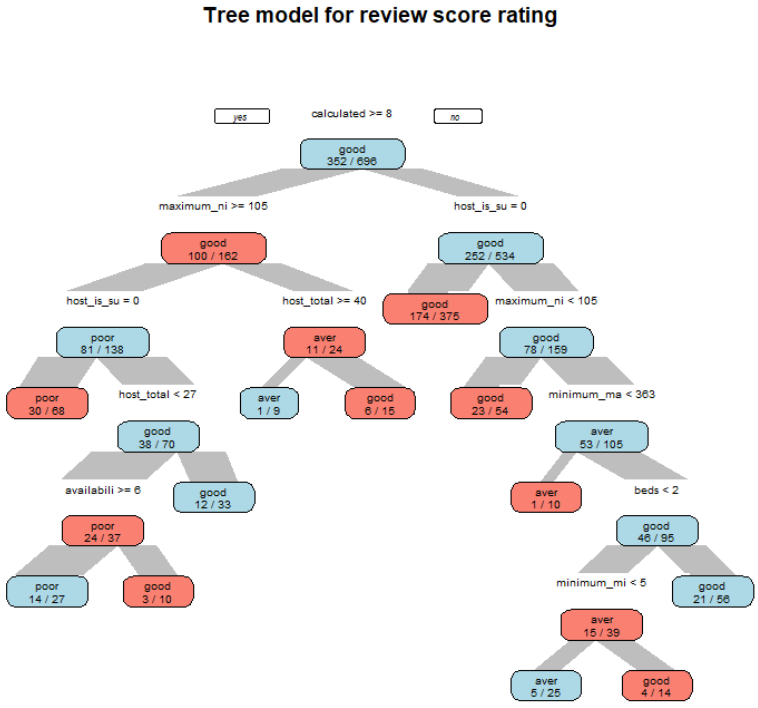
Tree Plot

```
# plotting the tree model

rpart.plot(tree, type = 2, extra = 110,box.col = c("lightblue", "salmon"))
```



Second Tree plot with parameters

```
# Same tree with different parameters

prp(tree, type = 1, extra = 3, split.font = 1, varlen = -4, fallen.leaves = FALSE , cex = 0.4,
    branch.lty = 2, box.col = c("lightblue", "salmon"), branch.type = 3, branch.lwd = 1.5,
    main = "Tree model for review score rating")
```

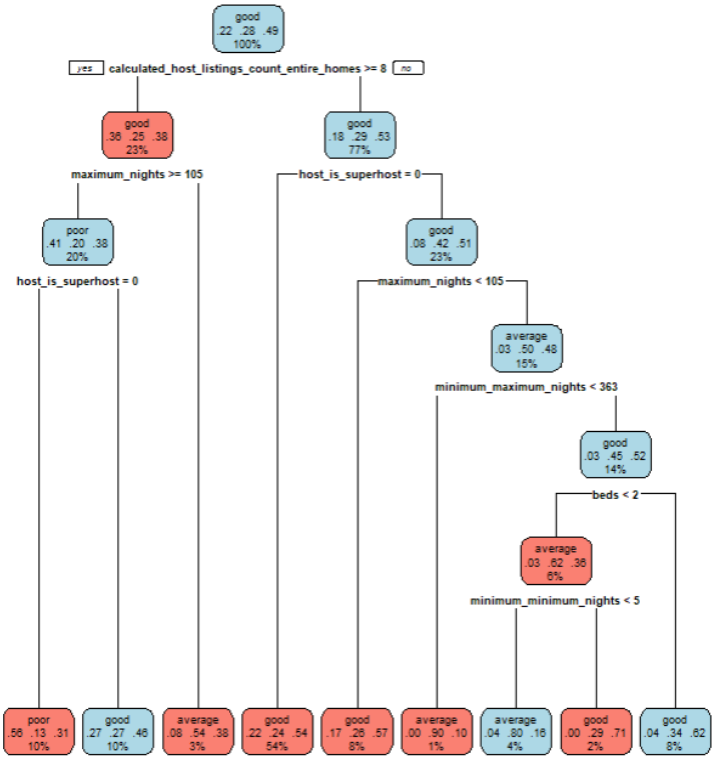


Now I have plotted the second tree with our cp value as follows:

```
tree2<-rpart(review_scores_rating~.,method="class",data=traintree,cp=0.0127,minsplit=0)
tree2
```

```
n= 696

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 696 352 good (0.22270115 0.28304598 0.49425287)
    2) calculated_host_listings_count_entire_homes>=7.5 162 100 good (0.36419753 0.25308642 0.38
271605)
      4) maximum_nights>=105 138  81 poor (0.41304348 0.20289855 0.38405797)
        8) host_is_superhost< 0.5 68  30 poor (0.55882353 0.13235294 0.30882353) *
        9) host_is_superhost>=0.5 70  38 good (0.27142857 0.27142857 0.45714286) *
      5) maximum_nights< 105 24  11 average (0.08333333 0.54166667 0.37500000) *
    3) calculated_host_listings_count_entire_homes< 7.5 534 252 good (0.17977528 0.29213483 0.52
808989)
      6) host_is_superhost< 0.5 375 174 good (0.22400000 0.24000000 0.53600000) *
      7) host_is_superhost>=0.5 159  78 good (0.07547170 0.41509434 0.50943396)
       14) maximum_nights< 105 54  23 good (0.16666667 0.25925926 0.57407407) *
       15) maximum_nights>=105 105  53 average (0.02857143 0.49523810 0.47619048)
         30) minimum_maximum_nights< 362.5 10   1 average (0.00000000 0.90000000 0.10000000) *
         31) minimum_maximum_nights>=362.5 95  46 good (0.03157895 0.45263158 0.51578947)
           62) beds< 1.5 39  15 average (0.02564103 0.61538462 0.35897436)
            124) minimum_minimum_nights< 4.5 25   5 average (0.04000000 0.80000000 0.16000000) *
            125) minimum_minimum_nights>=4.5 14   4 good (0.00000000 0.28571429 0.71428571) *
           63) beds>=1.5 56  21 good (0.03571429 0.33928571 0.62500000) *
```

```
rpart.plot(tree2, type = 2,box.col = c("lightblue", "salmon"))
```



Confusion Matrix

```
tree.pred <- predict(tree,traintree,type = "class")
confusionMatrix(tree.pred, traintree$review_scores_rating)
```

```
Confusion Matrix and Statistics

          Reference
Prediction poor average good
   poor      51      19   25
   average    2      37    5
   good     102     141  314

Overall Statistics

               Accuracy : 0.5776
                 95% CI : (0.5399, 0.6146)
    No Information Rate : 0.4943
    P-Value [Acc > NIR] : 0.000006322

                  Kappa : 0.2405

 Mcnemar's Test P-Value : < 0.00000000000000022

Statistics by Class:

                     Class: poor Class: average Class: good
Sensitivity              0.32903        0.18782      0.9128
Specificity              0.91867        0.98597      0.3097
Pos Pred Value           0.53684        0.84091      0.5637
Neg Pred Value           0.82696        0.75460      0.7842
Prevalence               0.22270        0.28305      0.4943
Detection Rate           0.07328        0.05316      0.4511
Detection Prevalence     0.13649        0.06322      0.8003
Balanced Accuracy        0.62385        0.58689      0.6112
```

```
> c <- table(traintree$review_scores_rating, tree.pred)
> c
         tree.pred
          poor average good
  poor      51       2  102
  average   19      37  141
  good      25       5  314
```

Prediction based on validation set.

```
tree.pred <- predict(tree,validtree,type = "class")
confusionMatrix(tree.pred, validtree$review_scores_rating)
```

```
Confusion Matrix and Statistics

          Reference
Prediction poor average good
   poor      23      17   33
   average    9      17    8
   good      74      75  209

Overall Statistics

               Accuracy : 0.5355
                 95% CI : (0.489, 0.5815)
    No Information Rate : 0.5376
    P-Value [Acc > NIR] : 0.5559

                  Kappa : 0.1287

 Mcnemar's Test P-Value : 0.000000000000001403

Statistics by Class:

                     Class: poor Class: average Class: good
Sensitivity              0.21698        0.15596      0.8360
Specificity              0.86072        0.95225      0.3070
Pos Pred Value           0.31507        0.50000      0.5838
Neg Pred Value           0.78827        0.78654      0.6168
Prevalence               0.22796        0.23441      0.5376
Detection Rate           0.04946        0.03656      0.4495
Detection Prevalence     0.15699        0.07312      0.7699
Balanced Accuracy        0.53885        0.55411      0.5715
```

In this I have initially crated the data which is approximately appropriate as per the review score rating and then I have binned the value in three different terms that are connected with the rating scores and I believe that grouping review score rating variables into discrete categories such as (“poor”, “average”, “good”) can simplify the classification tree model by reducing the number of splits needed to accurately predict the outcome variable and this can help avoid overfitting the model to the training data, and make the tree easier to interpret and communicate to others.

HoIver, I find that ideal size of the tree I have used the cross-validation process to identify and with the process I have summary of the results by looping 10 iterations each time building a new classification tree model with a different complexity parameter and the I thought that utilizing the which.min function would be great choice to find the row of the cp table attributes of the model object with the loIst cross-validation error rate and I find that furthermore, I have plot the tree with 2 different styles, as second one denotes extra parameters that can visualize the splits and the finally our model was able the capture the accuracy of 57.76% as the model describes that most of the review scores rating falls under good which means that many of them like it and second 197 are under average category and least under poor category with 155 through which I can consider that rentals have decent level of reviews rating in terms of score.

Step IV: Clustering

K-means analysis is a type of unsupervised machine learning algorithm that is used for clustering data. In K-means analysis, a dataset is divided into k clusters, where k is a predetermined number of clusters that is specified by the user. The goal of K-means analysis is to assign data points to clusters such that the data points in the same cluster are as similar as possible to each other and as different as possible from the data points in other clusters.

The result of K-means analysis is a set of k clusters, each containing a group of data points that are similar to each other. The similarity betIen data points within a cluster is typically measured using a distance metric such as Euclidean distance.

K-means analysis can be applied to a variety of data types and is commonly used in fields such as marketing, biology, and finance. Some examples of applications of K-means analysis include customer segmentation, image segmentation, and anomaly detection.

R script and result

```
#K means clustering --------------

#step 1 - creating the new data frame from the existing data my_data

k_means <- my_data[,c(17,18,35,38,41,42,43,44,45,46,47,48,49)]
view(k_means)
```

The code k_means <- my_data[,c(17,18,35,38,41,42,43,44,45,46,47,48,49)] selects a subset of columns from a data frame called my_data and assigns them to a new data frame called k_means.

The, c(17,18,35,38,41,42,43,44,45,46,47,48,49)] portion of the code specifies which columns to select. The c() function creates a vector of column indices that are to be included in the new data frame. In this case, columns 17, 18, 35, 38, 41, 42, 43, 44, 45, 46, 47, 48, and 49 are included in the new data frame.

```
#Step 2 finding the corelation between the numeric values
cor(k_meansss[,c(6:13)])
```

The [,c(6:13)] portion of the code specifies which columns to include in the correlation analysis. The c() function creates a vector of column indices, and the : operator creates a sequence of integers from 6 to 13, which includes columns 6, 7, 8, 9, 10, 11, 12, and 13 in the data frame.

The correlation coefficient is a measure of the strength and direction of the linear relationship betIen two variables. It ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 indicating no correlation. A positive correlation coefficient indicates that as one variable increases, the other variable tends to increase as Ill, while a negative correlation coefficient indicates that as one variable increases, the other variable tends to decrease.

```
> cor(k_meansss[,c(6:13)])
                            bedrooms       price minimum_nights maximum_nights minimum_minimum_nights
bedrooms                1.000000000  0.50981630   -0.004565582    0.004771875           -0.008820071
price                   0.509816295  1.00000000    0.033238933    0.020868908            0.033717688
minimum_nights         -0.004565582  0.03323893    1.000000000    0.088926902            0.997988155
maximum_nights          0.004771875  0.02086891    0.088926902    1.000000000            0.088589404
minimum_minimum_nights -0.008820071  0.03371769    0.997988155    0.088589404            1.000000000
maximum_minimum_nights -0.004905478  0.03478950    0.999560953    0.088006779            0.997638670
minimum_maximum_nights -0.053599775 -0.03208130    0.039596654    0.626972481            0.038487549
maximum_maximum_nights -0.052384963 -0.01551912    0.037021828    0.623611551            0.035943693
                       maximum_minimum_nights minimum_maximum_nights maximum_maximum_nights
bedrooms                       -0.004905478           -0.05359978           -0.05238496
price                           0.034789496           -0.03208130           -0.01551912
minimum_nights                  0.999560953            0.03959665            0.03702183
maximum_nights                  0.088006779            0.62697248            0.62361155
minimum_minimum_nights          0.997638670            0.03848755            0.03594369
maximum_minimum_nights          1.000000000            0.03872930            0.03884683
minimum_maximum_nights          0.038729300            1.00000000            0.98067008
maximum_maximum_nights          0.038846827            0.98067008            1.00000000
> 
```

Price has a moderate positive correlation with bedrooms (0.51), meaning that as the number of bedrooms in a listing increases, the price tends to increase as Ill.

minimum_nights and maximum_nights have a Iak positive correlation with price (0.03 and 0.02, respectively), suggesting that longer minimum or maximum booking requirements do not necessarily correspond to higher prices.

minimum_minimum_nights and maximum_minimum_nights have a strong positive correlation with minimum_nights (0.998 and 0.999, respectively) since they are essentially duplicates of minimum_nights.

minimum_maximum_nights and maximum_maximum_nights have a strong positive correlation with maximum_nights (0.999 and 1.00, respectively) since they are essentially duplicates of maximum_nights.

minimum_maximum_nights and maximum_maximum_night also have a strong positive correlation with each other (0.981), which is expected given that they are both related to the maximum booking requirement.

SCALLING

```
#Step 3 -Scalling
km.norm <- sapply(k_means[,c(6:13)], scale)
view(km.norm)
```

This code standardizes (or normalizes) the columns of the data frame "k_means" that have indices ranging from 6 to 13 (inclusive) using the scale() function, and stores the resulting standardized matrix as a new variable "km.norm".

Scaling (or normalizing) the variables in a dataset is a common preprocessing step before performing clustering analysis, and it is done for several reasons:

- Variables with larger scales tend to have a greater impact on the clustering results than variables with smaller scales. By scaling the variables, I ensure that all variables are equally important in determining the cluster assignments.
- Clustering algorithms are often based on the distance or similarity betIen the data points. If the variables are not scaled, then variables with larger scales will dominate the distance calculations and make the clustering results biased towards those variables.
- Scaling improves the interpretation of the clustering results by making the variables comparable in terms of their impact on the clustering assignments.

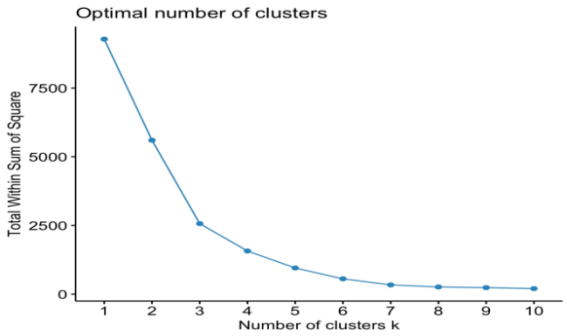
CREATING THE ELBOW CHART

An elbow chart is a common tool used to determine the optimal number of clusters for a given dataset when performing k-means clustering analysis. It is based on the idea that as the number of clusters increases, the within-cluster sum of squares (WCSS) typically decreases. The elbow point on the chart represents the number of clusters where the reduction in WCSS begins to level off, indicating that the benefits of increasing the number of clusters are diminishing.

```
#Step 4 creating the elbow chart
#installing the libraries factoextra and cluster to to make the elbow chart
#and do the further analysis

library(factoextra)
library(cluster)

elbow <- fviz_nbclust(km.norm,kmeans,method="wss",nstart=130)
elbow
```

For creating the elbow chart I have downloaded the factoextra library ,

After Analyzing the graph I have decided to take our K=4 and based on that I have created the further Analysi




```
#step 5
#Making the clusters
k1 <- kmeans(km.norm,centers = 4, nstart = 130)
k1$centers
```

This code performs K-means clustering on the normalized data km.norm using the kmeans() function with centers = 4 indicating that I want to divide the data into four clusters, and nstart = 130 specifying the number of initial random starting configurations to be tested.

The result of the clustering is saved in k1, which is a list containing various information about the clustering process such as the cluster assignments for each data point, the within-cluster sum of squares, etc.

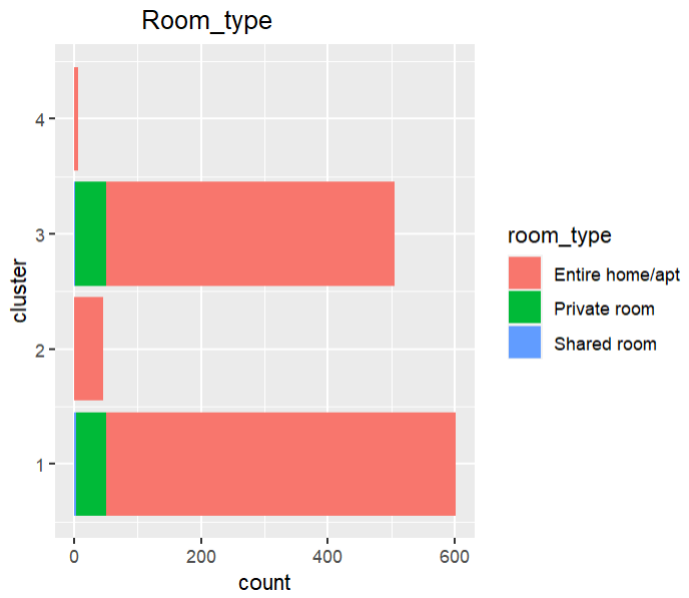
Finally, the code extracts the cluster centroids using the $centers attribute of the k1 object. These are the representative points for each cluster and can be used to interpret the characteristics of the clusters.

Results

```
> k1 <- kmeans(km.norm,centers = 4, nstart = 130)
> k1$centers
  minimum_nights maximum_nights minimum_minimum_nights maximum_minimum_nights minimum_maximum_nights
1    19.22168190      1.3620233            19.26284148            19.22158514              0.9006995
2    -0.09133533      0.5325092            -0.09243745            -0.09108418              0.8942434
3    -0.10586987     -0.6498057            -0.10525781            -0.10606230             -1.0575889
4     3.66091046      0.5731638             3.67456292             3.65826124              0.2183665
  maximum_maximum_nights minimum_nights_avg_ntm maximum_nights_avg_ntm
1              0.8886068            19.22837302              0.8973412
2              0.8859975            -0.09114094              0.8946233
3             -1.0472756            -0.10615213             -1.0577061
4              0.2020000             3.66168298              0.2101140
> |
```

DATA VISUALIZATION

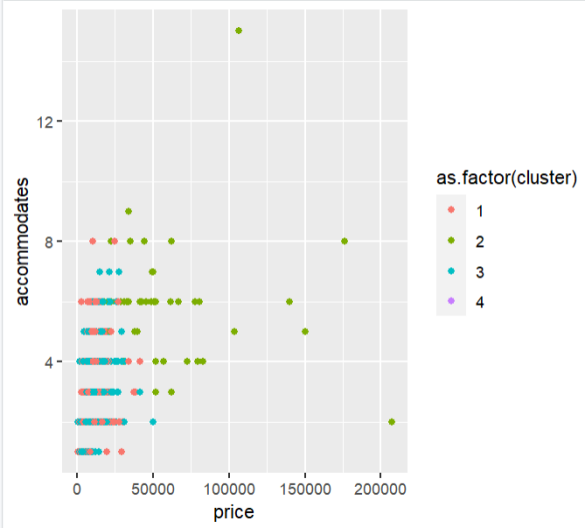
```
ggplot(data = k_means, aes(y = cluster)) +geom_bar (aes(fill = room_type)) +ggtitle("Room_type") +theme(plot.title = element_text(hjust = 0.3))
```



- Here I have created the cluster count for Room type , Here I have 3 points Entire home /apt,
  Private room /shared room

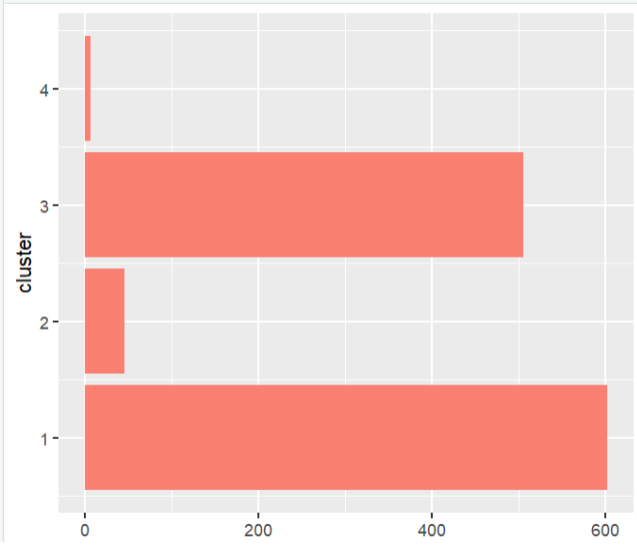
- Highest count is of entire home/ap then private room and lease shared room

```
ggplot(k_means, aes(x =price, y = accommodates, color = as.factor(cluster))) + geom_point()
```



In this visualization I can interpret that there are four different clusters in which second one is most expensive in terms of price with simultaneously less accommodates and on other hand first and first and third are less in terms of price with more number of accommodates and this connection can be very helpful for any manager to create the cost analysis and strategies based on cluster distribution.

```
# Count of Clusters by price

ggplot(k_means, aes(y = cluster)) +geom_bar(aes(color = price), fill = "salmon") +
  theme(plot.title = element_text(hjust = 0.3))
```



In this visualization I can interpret that the price is max with the cluster one, folloId by third cluster and least with the cluster four, which might mean that price fluctuations happens as per the season of Iather, Ieding, tournament and summer vacation as the demand is high on that period of time.

```r
k_means2 <- k_means[,c(4,5,6)]
k_means2$cluster <- k_means2$Cluster %>% as.factor()
colnames(k_means2)
```

```
> colnames(k_means2)
[1] "accommodates" "amenities"    "bedrooms"
>
```

Conclusion

In this whole project, analysis based on rentals at Belgrano, Buenos Aires in which I have undertaken several process and algorithms to understand the data and transform into valuable information with the help of visualizations, data interpretation and I found that entire home_apartment are the most liked room type as compared to private and shared rooms which means that new startups of hotel, Airbnb could offer entire home apartment at relatively high price at peak seasons as now with the help of data mining analysis they know the customer preference and behavior and create strategies accordingly furthermore, I also noticed that accommodates, bedrooms and availability is highly influencing the price of rentals which means the customer are strongly concerned about the quality, quantity, area of accommodates and timely booking as Belgrano is major tourist attraction, so management should aware about such trends which also connects the review score rating and most of them are good in terms of high rating which are associated to the accommodation and amenities in the rentals.

The insights extracted from the naïve bayes also indicates that people are highly concerned about decent living standards in terms of accommodation with four and want entire condo with entire home and apartment this means that people are not concerned much with the pricing cost and budget as it is small town, part from local's tourist generally come from long way to spend quality times with friends and family.

Finally in clustering, I observed an common trend that cluster one is most strongest one as compared to others as it offers more number of accommodates with better quality at cheap price and that is the biggest factor to attract the customer, because as an rational customer they want to enjoy the offering or product with minimum cost and resources and in many places that's what happen but the biggest enemy is competition, as they try everything to crush you in terms of making money and maintain healthy relationship with customer.