Data Mining for Business Analytics

Final Project - Retiro

Professor Page

May 8, 2023

Luke Brady, Paul Seiters, Elle Pencer, Jason Teng, and Wancheng Zhang

Table of Contents

Data Preperation and Explorations

1.1: Missing Value

This code reads in a CSV file of Airbnb listings in Buenos and creates a cleaned version of the data for the neighborhood of Retiro. The data cleaning includes removing columns that are not useful, filling in missing numeric values with the median, and filling in missing text values with placeholder text. The data frame is the final cleaned data set. The command is used to count the number of missing values in the cleaned data set.

```
missing_values_table <- data.frame(colSums(is.na(r)))
missing_values_table
percent_missing_table <- data.frame(colMeans(is.na(r))*100)
percent_missing_table
percent_missing <- colMeans(is.na(r)) * 100
variables_over_50 <- names(percent_missing[percent_missing > 50])
percent_missing_50_table <- data.frame(variables = variables_over_50, percent_missing = percent_missing[variabl
percent_missing_50_table
cols_to_remove <- names(percent_missing[percent_missing > 50])
r <- r[, !(names(r) %in% cols_to_remove)]
```

1.2: Summary Statistics

The code starts by using the lapply() function to apply a function to each element in the price column of the df_clean dataframe. The function converts the result to a numeric value.

```
df_clean$price <- lapply(df_clean$price, function(x) {
  x <- gsub("\\$|,", "", x)
  as.numeric(x)
})
```

The mean(), median(), sd(), min(), max(), and quantile() functions are then used to calculate various summary statistics for the price column.

```
> mean(df_clean$price)
[1] 14324.06
> median(df_clean$price)
[1] 8283
> sd(df_clean$price)
[1] 78122.27
> min(df_clean$price)
[1] 1100
> max(df_clean$price)
[1] 2300811
```

Mean and sd values indicate that the values in the price column are quite spread out from the mean, with some high value outliers.

Based on the summary statistics calculated for the price column, we can make a few assumptions about the data:
The mean value of $14,324.06 is significantly higher than the median value of $8,283, which suggests that the distribution of values in the price column is skewed towards higher values. This is further supported by the high standard deviation of $78,122.27. Presence of outliers: The minimum value of $1,100 and the maximum value of $2,300,811 are quite far apart, indicating the presence of outliers in the price column. The standard deviation is also quite high, which is another indication of the presence of outliers.

1.3: Data Visualization

Our first data visualization is a bar plot that looks at the beds vs bedrooms in our neighborhood group Retiro. We used a bar plot because it allows us to easily compare the number of beds and bedrooms in each listing and understand the accommodation capacity. This information is important in evaluating the demand for the listing, its rental price, and its overall performance on the platform. Additionally, the bar plot helps group travelers determine if they will have private space or if they will have to share a bed. Finally, it also helps us understand why rental rates vary among listings since the number of bedrooms and beds can impact the rental price.
Plot 2 is a bubble plot that shows the review of cleanliness vs reviews per month. We chose this type of plot because it allows us to visualize the relationship between two variables and identify patterns in the data. The cleanliness of an apartment is a crucial factor in the rental experience, so we wanted to see if there was a correlation between the review score and the cleanliness of the apartment. A high number of negative reviews related to cleanliness could deter potential guests from staying at the property, while positive reviews could encourage them to book. Plot 3 is a scatter plot that looks at price vs review score. We used a scatter plot because it's an effective way to visualize the relationship between two continuous variables. The review score is an indicator of the quality of the rental, including its cleanliness, amenities, and overall condition. By comparing the review score to the price, we can assess whether the rental offers good value for its quality. The scatter plot also gives us a better understanding of the experience that they will be receiving and the quality that they can expect.

Plot 4 looks at how many people can be accommodated vs price. We used a bar plot to compare the number of rooms and beds to the rental price. This information is helpful in determining the ideal price for a rental and why it varies. The bar plot shows us that people typically look for listings with 2-4 rooms and compare them to the price. Plot 5 looks at the proportion of room types in Retiro. We used a pie chart because it allows us to easily see the proportion of different room types available in the neighborhood. This information is important in understanding the rental market and what type of homes are typically offered. The majority of the room types

available are the entire home/apartment, which gives us insight into what people are typically looking for in the rental market.

1.4: Mapping

Step Mapping generates a map of the Airbnb listings in the dataset, with each listing represented by a circle at its corresponding geographic coordinates. It can be useful for getting a quick visual understanding of the spatial distribution of listings in the dataset. And it also shows that there is a record located far from the group.

1.5: Worldcloud

The word cloud displays several frequently occurring words such as Buenos Aires, San, Barrio, Ciudad, Recoleta, Plaza, Restaurants, Shopping, City, Madero, Cultural, and more. From this word cloud, we can see that the words are related to Buenos Aires and its various neighborhoods, attractions, and cultural offerings. Additionally, the word cloud indicates that Buenos Aires is a shopping city with several restaurants, making it a desirable destination for tourists who want to explore the city's culture and indulge in various activities.

Prediction

2.1: Multiple Regression Model

After reading the CSV file in R, there are thousands of items and 75 different variables. In the beginning, I filtered the neighborhood to Retiro, our designed neighborhood. I found out the price column, host_response_rate, and host_acceptance_rate is the character, and those variables are important for my regression model, so I change those variables into numeric.

After changing the format of the price, I delete other non-numeric variables in the dataset. Since some other numerical variables are not useful such as ID number and latitude, I also removed those columns.

To fill up those NAs, I used the median because the data set I use is not normal distribution. Replacing missing values with the median helps to maintain the overall shape and characteristics of the original data. Those missing values I filled up by median are rounded to integers because some variables can’t be with decimal points like bedrooms and beds.

I also use the log function in price because the range is too large. If we don’t use the log function, it can lead to heteroscedasticity, which means that the variability of the response variable changes as a function of the predictor variables. Heteroscedasticity can lead to biased estimates of the regression coefficients and inaccurate predictions.

Applying a log transformation to the response variable can help to reduce heteroscedasticity by compressing the range of the response variable. This can result in a more evenly distributed error term, which leads to more accurate and reliable predictions.

Next step, I made training set and testing set because the training set is typically used to fit the parameters of our model, such as coefficients in a regression model or weights in a neural network. We use the training set to adjust the model to the patterns in the data, so that it can accurately predict outcomes for new data points.

After split the data set into training set and testing test, I started to do multiple linear regression. I chose review host response rate, host total listings count, accommodates, bedrooms, beds, minimum night, availability 30, review scores cleanliness, calculated host listings count, calculated host listing count entire homes, calculated host listings count private rooms, reviews per month, review scores value, review scores check-in, review scores rating, availability 60, and availability 90 as independent variables because I think those variables can affect the price. I think the quality of my model is great because every function I use can make sure the p-values are within the significant value.

Result of Regression:

```
Call:
lm(formula = price ~ host_response_rate + host_total_listings_count +
    accommodates + bedrooms + beds + minimum_nights + availability_30 +
    review_scores_cleanliness + calculated_host_listings_count +
    calculated_host_listings_count_entire_homes + calculated_host_listings_count_private_rooms +
    reviews_per_month + review_scores_value + review_scores_checkin +
    review_scores_rating + availability_60 + availability_90,
    data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5642 -0.2692 -0.0591  0.2149  5.4433
```

```
Coefficients:
                                             Estimate Std. Error t value Pr(>|t|)
(Intercept)                                 8.4340973  0.4322812  19.511  < 2e-16 ***
host_response_rate                         -0.0028331  0.0014808  -1.913 0.056281 .
host_total_listings_count                   0.0012415  0.0007356   1.688 0.092074 .
accommodates                                0.1778670  0.0301893   5.892 6.91e-09 ***
bedrooms                                    0.3215787  0.0482229   6.669 6.67e-11 ***
beds                                       -0.0669049  0.0313722  -2.133 0.033428 *
minimum_nights                             -0.0028166  0.0008629  -3.264 0.001171 **
availability_30                             0.0184623  0.0058127   3.176 0.001582 **
review_scores_cleanliness                   0.1971677  0.0882025   2.235 0.025819 *
calculated_host_listings_count              0.6783190  0.2033194   3.336 0.000911 ***
calculated_host_listings_count_entire_homes -0.6761329  0.2033429  -3.325 0.000947 ***
calculated_host_listings_count_private_rooms -0.7038466  0.2032715  -3.463 0.000580 ***
reviews_per_month                          -0.0645638  0.0162270  -3.979 7.92e-05 ***
review_scores_value                        -0.3673739  0.1063443  -3.455 0.000597 ***
review_scores_checkin                      -0.2795924  0.1036881  -2.696 0.007237 **
review_scores_rating                        0.4792658  0.1210992   3.958 8.63e-05 ***
availability_60                            -0.0114682  0.0052739  -2.175 0.030121 *
availability_90                             0.0047883  0.0026111   1.834 0.067263 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4933 on 515 degrees of freedom
Multiple R-squared:  0.4398,    Adjusted R-squared:  0.4213
F-statistic: 23.78 on 17 and 515 DF,  p-value: < 2.2e-16
```

The residual standard error (RSE) of 0.4933 represents the standard deviation of the errors made by the model in predicting the outcome variable. This value is relatively small, which indicates that the model is able to make reasonably accurate predictions. The multiple R-squared value is 0.4398 which indicates that approximately 44% of the variance in the response variable can be explained by the predictor variables. The adjusted R-squared value is 0.4213 which is slightly smaller than the multiple R-squared value, suggesting that some of the variables in the model may not be contributing significantly to the explanation of the response variable. The F-statistic of 23.78 and its associated p-value indicate that the overall model is statistically significant and provides a better fit than the null model.

Overall, while the model appears to be statistically significant, the relatively low R-squared and adjusted R-squared values suggest that there may be room for improvement in the model's performance. Additionally, further evaluation of the model's assumptions, such as linearity, normality, and homoscedasticity, should be conducted to ensure that the model is appropriate for the data.

```{r}
accuracy(log_pred1, train$price)
```

```
                     ME      RMSE       MAE         MPE     MAPE
Test set -5.572466e-15 0.4848868 0.331834 -0.2524781 3.565775
```

```{r}
accuracy(log_pred2, test$price)
```

```
                   ME      RMSE       MAE         MPE     MAPE
Test set -0.03998023 0.4152738 0.3068993 -0.6296879 3.339178
```

First set (Training set):
- ME (Mean Error): The mean error in this case is approximately -5.57e-15, which indicates that, on average, the predictions on the training set are extremely close to the actual values. The ME being close to zero suggests that the model is not consistently overestimating or underestimating the targets in the training set.
- RMSE (Root Mean Squared Error): The root mean squared error is 0.4848868, indicating the average magnitude of the residuals (the differences between the predicted and actual values) in the training set. Lower RMSE values indicate better predictive performance.
- MAE (Mean Absolute Error): The mean absolute error is 0.331834, which represents the average absolute difference between the predicted and actual values in the training set. Similar to RMSE, lower MAE values suggest better accuracy.
- MPE (Mean Percentage Error): The mean percentage error is approximately -0.2524781%. This metric measures the average percentage difference between the predicted and actual values in the training set. A negative value indicates an overall underestimation, while a positive value would indicate an overestimation.
- MAPE (Mean Absolute Percentage Error): The mean absolute percentage error is 3.565775%. It measures the average absolute percentage difference between the predicted and actual values in the training set. A lower MAPE indicates better accuracy.

Second set (Testing set):
- ME (Mean Error): The mean error is approximately -0.03998023, indicating a slight underestimation bias in the predictions on the testing set. However, the ME is close to zero, suggesting that, on average, the model's predictions are reasonably accurate.

- RMSE (Root Mean Squared Error): The root mean squared error is 0.4152738, indicating the average magnitude of the residuals in the testing set. Lower values suggest better predictive performance.
- MAE (Mean Absolute Error): The mean absolute error is 0.3068993, representing the average absolute difference between the predicted and actual values in the testing set. Lower values indicate better accuracy.
- MPE (Mean Percentage Error): The mean percentage error is approximately -0.6296879%. Similar to the training set, a negative value suggests an overall underestimation.
- MAPE (Mean Absolute Percentage Error): The mean absolute percentage error is 3.339178%. It measures the average absolute percentage difference between the predicted and actual values in the testing set. Lower MAPE values indicate better accuracy.

Overall, these metrics provide information about the performance of the model on the training and testing sets. The model appears to perform well on the training set, with low RMSE and MAE values. However, it slightly underestimates the targets in the testing set, as indicated by the negative ME and MPE values. The RMSE and MAE values for the testing set are also relatively low, indicating reasonably accurate predictions.

Classification

3.1: K-nearest Neighbors

For the K-nearest model, we started by cutting the data frame to the numerical predictors; review scores rating, accuracy, cleanliness, checkin, communication, location, value, and our eventual output under amenities, "wifi." We then converted wifi to a binary output of "0" being no wifi and "1" as having wifi in the accommodation. From there, we cut the amenities column since we had created the binary wifi column and established a data partition splitting the data 60:40. To arrive at the predictors, we discussed the most likely predictors correlating to having wifi, with a customer's overall perception of accommodation. For example, if a place is clean, has an overall high rating, or is an optimal location, one could infer the quality of accommodation is high, and they would have a basic amenity like wifi. So, we chose review score predictors and them through a welch two-sample t-test to gauge p-value, and removed the variables with a p-value greater than 0.05 as it is not statistically significant and could harm the model; the only variable we removed was review_scores_location. From there, we take the values from "Apartamento de 1 cuarto cerca de Plaza San Martín" and preProcess the data while adjusting the k-value to mitigate "noise" and stop overfitting; we arrived at a k-value of 7. Finally, we arrived at the seven nearest neighbors and can gather their index values; 241, 285, 361, 158, 165, 217, and 11.

3.2: Naive Bayes

Selected Variables

```{r}
r_clean_NB <- select(r_clean, review_scores_rating, review_scores_accuracy, review_scores_cleanliness,
review_scores_checkin, review_scores_communication, review_scores_location, review_scores_value, price, beds,
accommodates, host_is_superhost, host_response_time, calculated_host_listings_count, host_has_profile_pic,
host_identity_verified, number_of_reviews, reviews_per_month, instant_bookable)
```

```{r}
table(r_clean_NB$instant_bookable)
```

```
  f   t
587 301
```

Make price numeric

```{r}
r_clean_NB$price <- gsub("[\\$,]", "", r_clean_NB$price)
r_clean_NB$price <- as.numeric(r_clean_NB$price)
```

Turn categorical variables into factors

```{r}
r_clean_NB$host_is_superhost <- factor(r_clean_NB$host_is_superhost)
r_clean_NB$host_response_time <- factor(r_clean_NB$host_response_time)
r_clean_NB$instant_bookable <- factor(r_clean_NB$instant_bookable)
r_clean_NB$host_has_profile_pic <- factor(r_clean_NB$instant_bookable)
r_clean_NB$host_identity_verified <- factor(r_clean_NB$instant_bookable)
r_clean_NB$beds <- factor(r_clean_NB$beds)
r_clean_NB$accommodates <- factor(r_clean_NB$accommodates)
```

Equal frequency binning because of multiple very skewed ratings

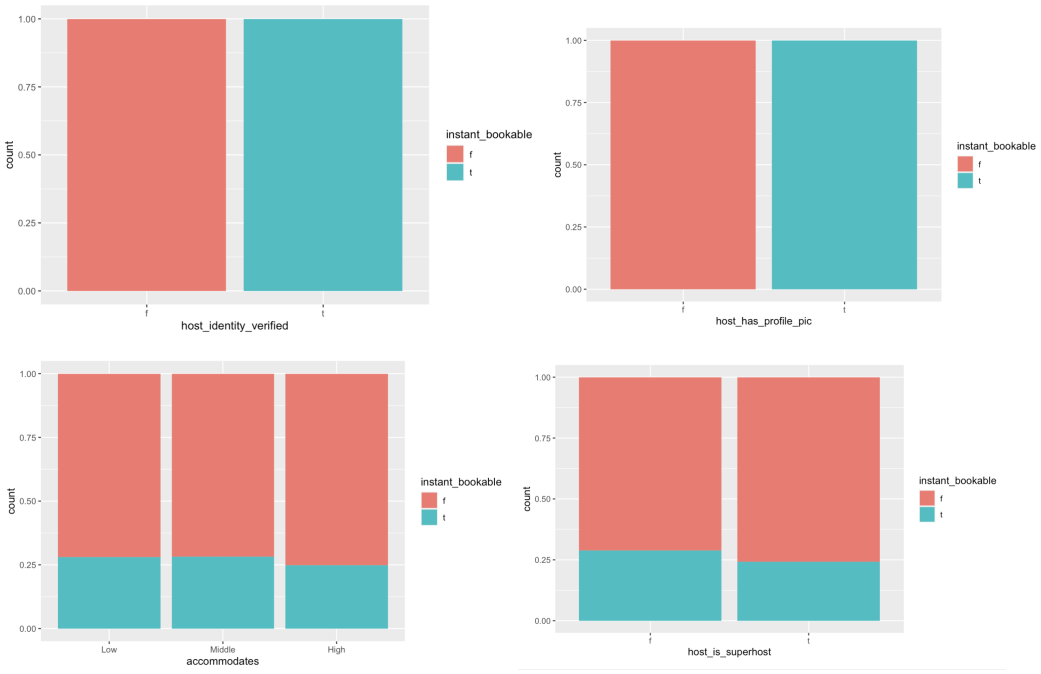
```{r}
vars_to_bin <- c("review_scores_rating", "review_scores_accuracy",
                 "review_scores_cleanliness", "review_scores_checkin",
                 "review_scores_communication", "review_scores_location",
                 "review_scores_value", "price",
                 "host_response_time", "calculated_host_listings_count",
                 "number_of_reviews", "reviews_per_month", "beds", "accommodates")

for (var in vars_to_bin) {
  r_clean_NB[[var]] <- as.factor(ntile(r_clean_NB[[var]], 3))
}


levels(r_clean_NB$review_scores_accuracy) <- c("Low", "Middle", "High")
levels(r_clean_NB$review_scores_cleanliness) <- c("Low", "Middle", "High")
levels(r_clean_NB$review_scores_checkin) <- c("Low", "Middle", "High")
levels(r_clean_NB$review_scores_communication) <- c("Low", "Middle", "High")
levels(r_clean_NB$review_scores_location) <- c("Low", "Middle", "High")
levels(r_clean_NB$review_scores_value) <- c("Low", "Middle", "High")
levels(r_clean_NB$price) <- c("Low", "Middle", "High")
levels(r_clean_NB$host_response_time) <- c("Low", "Middle", "High")
levels(r_clean_NB$calculated_host_listings_count) <- c("Low", "Middle", "High")
levels(r_clean_NB$number_of_reviews) <- c("Low", "Middle", "High")
levels(r_clean_NB$reviews_per_month) <- c("Low", "Middle", "High")
levels(r_clean_NB$beds) <- c("Low", "Middle", "High")
levels(r_clean_NB$accommodates) <- c("Low", "Middle", "High")
```

Data Partition

```{r}
library(dplyr)
set.seed(50)
sampler <- sample_n(r_clean_NB, 888)
r_clean_train <- slice(r_clean_NB, 1:533)
r_clean_valid <- slice(r_clean_NB, 534: 888)
```

Visualizations

```{r}
ggplot(r_clean_train, aes(x = review_scores_rating, fill = instant_bookable)) +
  geom_bar(position = "fill")

ggplot(r_clean_train, aes(x = review_scores_accuracy, fill = instant_bookable)) +
  geom_bar(position = "fill")


ggplot(r_clean_train, aes(x = review_scores_cleanliness, fill = instant_bookable)) +
  geom_bar(position = "fill")


ggplot(r_clean_train, aes(x = review_scores_checkin, fill = instant_bookable)) +
  geom_bar(position = "fill")



ggplot(r_clean_train, aes(x = review_scores_communication, fill = instant_bookable)) +
  geom_bar(position = "fill")


ggplot(r_clean_train, aes(x = review_scores_value, fill = instant_bookable)) +
  geom_bar(position = "fill")


ggplot(r_clean_train, aes(x = price, fill = instant_bookable)) +
  geom_bar(position = "fill")
```



Model

```
Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
       f        t
0.727955 0.272045

Conditional probabilities:
   review_scores_rating
Y           1         2         3
  f 0.2577320 0.4355670 0.3067010
  t 0.4827586 0.3586207 0.1586207

   review_scores_accuracy
Y         Low    Middle      High
  f 0.2680412 0.4639175 0.2680412
  t 0.4620690 0.3793103 0.1586207

   review_scores_cleanliness
Y         Low    Middle      High
  f 0.2860825 0.4432990 0.2706186
  t 0.4344828 0.3655172 0.2000000

   review_scores_checkin
Y         Low    Middle      High
  f 0.2654639 0.5206186 0.2139175
  t 0.5034483 0.3724138 0.1241379

   review_scores_communication
Y         Low    Middle      High
  f 0.2551546 0.5257732 0.2190722
  t 0.4896552 0.3655172 0.1448276

   review_scores_location
Y         Low    Middle      High
  f 0.2422680 0.5515464 0.2061856
  t 0.4758621 0.3655172 0.1586207

   review_scores_value
Y         Low    Middle      High
  f 0.2731959 0.3969072 0.3298969
  t 0.4620690 0.3793103 0.1586207

   price
Y         Low    Middle      High
  f 0.3015464 0.3453608 0.3530928
  t 0.4000000 0.3448276 0.2551724

   beds
Y         Low    Middle      High
  f 0.3865979 0.3298969 0.2835052
  t 0.4896552 0.2758621 0.2344828

   host_is_superhost
Y           f         t
  f 0.6365979 0.3634021
  t 0.6896552 0.3103448

   host_response_time
Y         Low    Middle      High
  f 0.5438144 0.4561856 0.0000000
  t 0.2758621 0.7241379 0.0000000

   calculated_host_listings_count
Y         Low    Middle      High
  f 0.4845361 0.2835052 0.2319588
  t 0.4206897 0.2620690 0.3172414

   number_of_reviews
Y         Low    Middle      High
  f 0.2371134 0.2989691 0.4639175
  t 0.1793103 0.2137931 0.6068966

   reviews_per_month
Y         Low    Middle      High
  f 0.5206186 0.3221649 0.1572165
  t 0.3586207 0.2965517 0.3448276
```

Training Set and Test Set Stats:

```
Confusion Matrix and Statistics

          Reference
Prediction   f   t
         f 309  74
         t  79  71

               Accuracy : 0.7129
                 95% CI : (0.6725, 0.751)
    No Information Rate : 0.728
    P-Value [Acc > NIR] : 0.7966

                  Kappa : 0.283

 Mcnemar's Test P-Value : 0.7464

            Sensitivity : 0.7964
            Specificity : 0.4897
         Pos Pred Value : 0.8068
         Neg Pred Value : 0.4733
             Prevalence : 0.7280
         Detection Rate : 0.5797
   Detection Prevalence : 0.7186
      Balanced Accuracy : 0.6430

       'Positive' Class : f
```

```
Confusion Matrix and Statistics

          Reference
Prediction   f   t
         f 147  96
         t  52  60

               Accuracy : 0.5831
                 95% CI : (0.5299, 0.6349)
    No Information Rate : 0.5606
    P-Value [Acc > NIR] : 0.2115125

                  Kappa : 0.1272

 Mcnemar's Test P-Value : 0.0004084

            Sensitivity : 0.7387
            Specificity : 0.3846
         Pos Pred Value : 0.6049
         Neg Pred Value : 0.5357
             Prevalence : 0.5606
         Detection Rate : 0.4141
   Detection Prevalence : 0.6845
      Balanced Accuracy : 0.5617

       'Positive' Class : f
```

Overall we had to overcome multiple obstacles to produce this model. First, we included all available categorical or numerical parameters in our initial data selection. The specific variables can be seen in the code. We performed preprocessing steps on price, which was in string format and also converted categorical variables (also still in string format) and numerical variables into factors. We then split our data into 40% test set and 60% training set. We visualized the training set portion and came to the conclusion that we have to remove variables that do not show any difference among outcome classes from the model. Most importantly, however, it was also imperative to remove the two variables profile picture and host identity variables, because their outcomes perfectly correlated with the outcome of "instant_bookable". If we would have included them our accuracy would have been 100% in the training data. We actually tested this and saw that the test set accuracy would also have been 100%.

When testing our model, we see that our Naive Bayes Model does not generalize well on unseen test data. In fact, the accuracy drops significantly from 74.6% to 58.3%, which is just slightly better than random. This is not surprising as Naive Bayes is prone to overfitting, especially when the training data is relatively small, which is the case for us with just 888 records to work with.

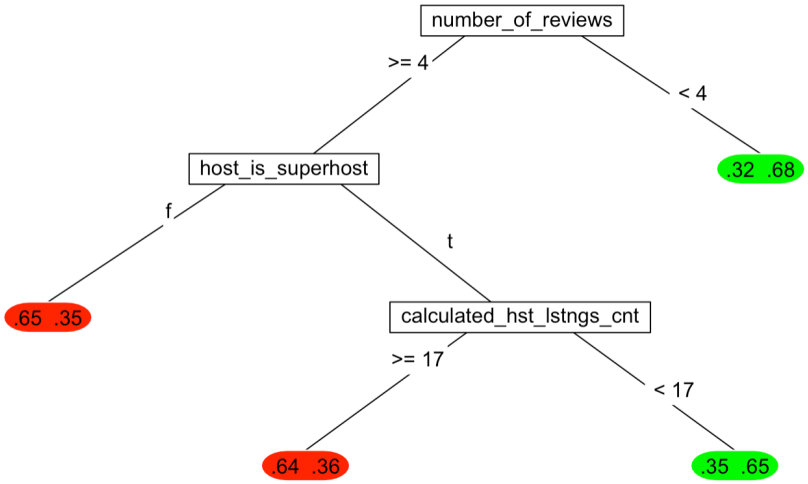
3.3 Classification Tree

For the classification tree, we considered all inputs that could contribute to the review scores rating of a person's stay at various accommodations. Some of the inputs we included are price, beds, host response time, number of reviews, and whether the accommodation could be instantly booked. The only road bump we encountered was converting the price from a character to a numerical value which was vital for the model. Once we completed this conversion, we factorized the price value and binned it into either "low score" or "high score," with a cutoff around the median value of 4.8, shown in the summary statistics. The next step in the classification tree was to data partition, using our standard method with the sample_n function. We then built our tree model, with the review scores rating as the output variable, setting a max depth of 4, allocating the color green to represent a high score and red for a low score. If we follow the tree down, we notice the first split is done on the number of reviews, either greater/equal to 4, followed by whether a host is a "super host," the number of reviews being greater/equal to 19, then on the final split of "calculated host listings count." The results showed the proportion of high vs. low values and where they fell based on the split among the four levels.

The next classification we generated showed the outcome as proportions and, again, was split on the number of reviews being greater/equal to four, and our first outcome on the right showed

.32/.68 meaning the model predicts a 68% chance of being a high rating. The rating had two other splits on whether the host is a "super host" and the "calculated host listings count." Between the first two classification trees, we were also able to conduct cross-validation, which gave us a root node error of 0.499, which is the percent of correctly sorted records at the first (root) splitting node.

On a final note, we can state the accuracy scores and generalize relatively well to unseen data. Although the accuracy score for the validation was 0.15% higher than the training data, there are multiple reasons for this occurrence, including being a small dataset; we only had 533 rows in our data, so the model may have had difficulty finding a good fit for the training data. Thus, returning a lower accuracy score on the training data and higher on the validation set.



Clustering

4.1: K-Means Analysis OR Hierarchical Clustering

Step IV Clustering starts by selecting a subset of columns from the original dataset. This subset of columns contains numerical variables that can be used for clustering.

Then, the kmeans() function is used to perform k-means clustering. The centers argument specifies the number of clusters to create, which is set to 5 in this case.

Next, the aggregate() function is used to calculate the mean of each variable within each cluster. This provides a summary of the characteristics of each cluster.

Cluster 1 has the highest average number of reviews which is 32.17, so this cluster is very popular.

Cluster 2 has the highest average accommodation which is 6.5, so this cluster is for big groups.

Cluster 3 has the highest average price which is 2300811, so this cluster is very expensive.

Cluster 4 has the highest average calculated_host_listings_count which is 26.31, so this cluster has many available choices.
Cluster 5 has the highest average minimum_nights which is 13, so this cluster is not for temporary plans.
After that, the ggplot2 library is loaded and three scatter plots are created to visualize the relationships between different variables and the assigned clusters. The first plot shows the relationship between the logarithm of price and accommodation, with each point colored by its assigned cluster. In general, clusters with big numbers (4,5) are more expensive and clusters with small numbers do not support big groups because of small accommodation.
The second plot shows the relationship between the logarithm of price and the number of reviews in the last 12 months, with each bar colored by its assigned cluster. Clusters with higher prices have fewer review numbers. And most records have around 50 reviews.
The third plot shows the relationship between the logarithm of price and the review score for communication, with each point colored by its assigned cluster. Most records have a 4.5 to 5 review score. Seems like price does not affect the score.
Overall, this R code performs k-means clustering on a subset of columns from a dataset of Airbnb listings, assigns each observation to a cluster, summarizes the characteristics of each cluster, and creates visualizations to help interpret the results.

Conclusions

5.1: Conclusion and Final Remarks

The Wizards conducted an extensive analysis of the Reitro Airbnb dataset, which included listings in Buenos, and discovered various outcomes by exploring multiple components related to the apartment, host, and overall listing. We used data mining techniques such as cleansing and calculated summary statistics to gain valuable insights. We also used different types of visualizations to understand the relationships between variables and to determine what factors the Airbnb users were considering when deciding on their room selection. Finally, we built a multiple regression model to predict the price of Airbnb listings using numeric variables and identified key factors that impact the price of Airbnb listings in Buenos.

We also explored several machine learning models, including K-nearest Neighbors, Naive Bayes, Classification Tree, and K-Means Clustering. We preprocessed the data, established predictors, and tuned the k-value to mitigate overfitting for the K-nearest Neighbors model. We trained the Naive Bayes model using all available categorical and numerical parameters and removed variables that showed no difference in outcome classes to improve accuracy. We used cross-validation to measure the accuracy of the Classification Tree model, and for the K-Means Clustering model, we selected a subset of columns, performed preprocessing steps, and chose the optimal number of clusters using the elbow method.

The results from these models could be used in the real world. For instance, Airbnb hosts could use the K-nearest neighbor model to determine the likelihood of a guest requiring wifi based on their review scores. Airbnb could use the Classification Tree model to identify listings that are likely to receive high review scores and promote them more prominently on their platform.

Lastly, Airbnb could use the K-Means Clustering model to identify clusters of listings that are popular with certain types of guests and target them with more personalized recommendations. Overall, our analysis provides valuable insights that can help travelers make informed and cost-effective decisions when choosing an Airbnb listing in Buenos.