Report: Predicting Airbnb Prices

Charlie Mei (cm3947)

1 Introduction

This report outlines the process undertaken to predict Airbnb rentals prices in New York. This project was based on a dataset consisting of 90 features related to the property, the host and reviews of over 35,000 rental properties. The goal was to develop a supervised machine learning model that resulted in the lowest root mean squared error (RMSE) on a previously unseen dataset of Airbnb rentals.

2 Project workflow

A first pass at tackling this project resulted in a number of complications:

- Using one script file for all code was cumbersome for code review, editing and debugging.
- Not commenting code meant that understanding code written on previous days was time consuming and sometimes undecipherable.
- Blindly testing models and not recording which features were used in each model run resulted in confusion as to what methods had already been tested.

To address these complications, a consistent project workflow was developed to complete this project. The workflow consisted of:

1. using separate script files for different coding purposes. Code was written in two separate files - one file for reading and cleaning data, the other file for data modeling.
2. organizing script files into sections, separated by comments. The first script file was called *A_CLEANING*. All packages used were placed at the top of the script. Data were then imported before cleaning. The second script file was called *B_MODELING*. Here, the workspace was first cleared before the cleaned data were loaded again to be used for modeling.
3. having a modeling framework to guide model tuning and development, as described below.

3 Modeling framework

The modeling framework used in this project is illustrated in Figure 1 overleaf.

The framework consists of four components: reading and tidying data, data transformation, modeling, and inference. Data were imported into R and tidied so that it was appropriate for modeling. Then, the cyclical and iterative process of data transformation, modeling, inference and analysis proceeded to find the optimal model that resulted in the lowest RMSE.

The first two components were scripted in the *A_CLEANING* file. The latter components were scripted in the *B_MODELING* file.

The processes undertaken in each of the four components are discussed below.

Reading and tidying data

Data were read into R before being split into a training and test set using the **caret** package. Seventy per cent of the data were allocated to the training dataset.

The features in the dataset were broken into five categories:

1. features about the host
2. features about the location of the rental property
3. features about the property itself
4. features about reviews on the property

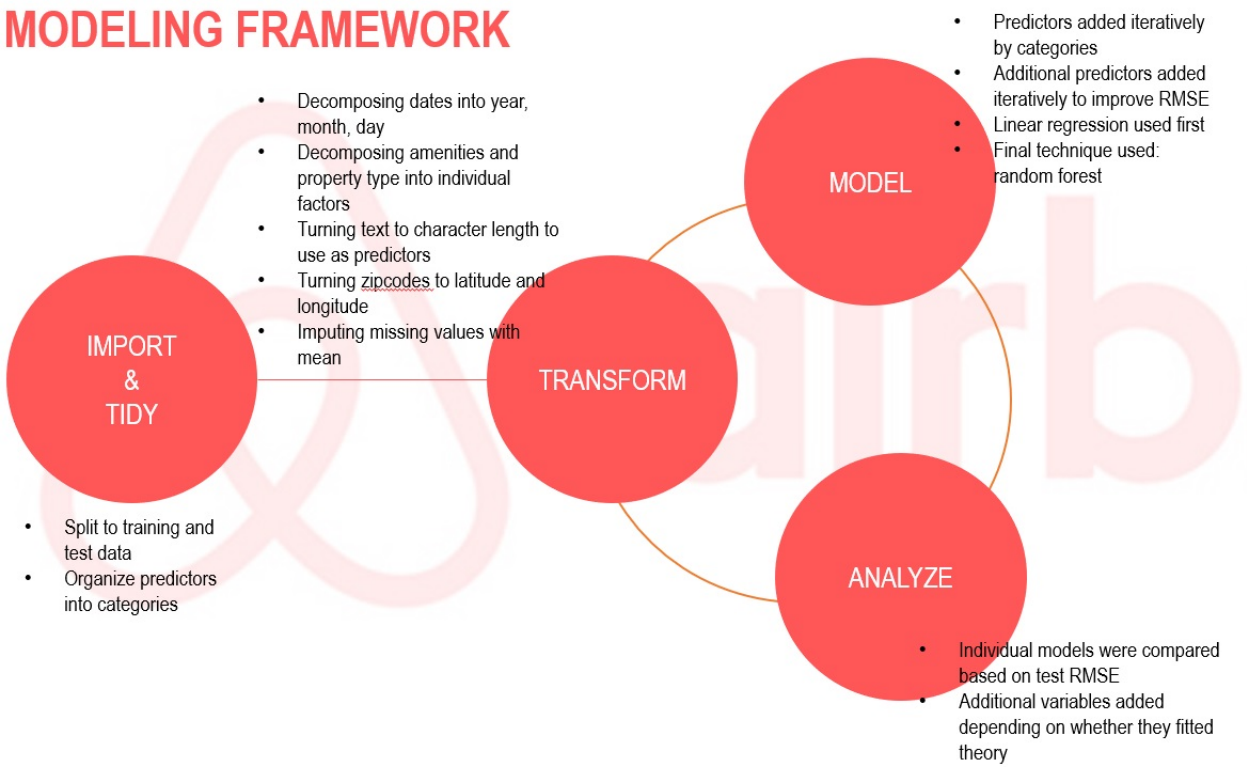

Figure 1: Modeling framework

5. other miscellaneous features.

In terms of tidying data, all missing values in the features were imputed using the mean. An alternative could have been to use the **mice** package to implement a more rigorous imputation strategy. However, without a thorough understanding of the imputation method used in **mice**, it was decided to proceed with the simpler method that was more easily interpretible.

In cases where there were only a small amount of missing values, the missing observations were dropped from analysis to avoid introducing bias from imputation. For example, the beds variable only had three missing observations and these three observations were removed from all further analysis.

Data transformation

Feature engineering was used to create meaningful additional features from text data in the original dataset. For each of the five categories, this involved the following:

- Features about the host: host since was turned into a date format and number of days from 1 January 2019, boolean values for verified host and superhosts were turned to factors.
- Features about location: zipcodes were turned to latitude and longitudes using the **zipcode** package
- Features about the property: amenities, property type, and cleaning fees, were broken down to individual factors on whether or not a property had the specific quality
- Features about reviews: first and last review days were turned to dates and number of days from 1 January 2019. The difference between first and last review days was also used as an indicator of how recently the property was reviewed.

Additional text analysis could have been useful to extract even more useful features for modeling. For example, constructing word clouds could provide additional information relating to what words were most commonly used in property descriptions, which can be used as predictors of prices.

Each of these features were added iteratively into the model, as described below.

Modeling

Initially, linear regression was used to correlate Airbnb rental prices with property features. Features were initially added by category; features about the host were first run, followed by adding in features about location, and so on. When running linear models, only non-feature engineered variables were used.

The process of iteratively including additional features by category resulted in a test RMSE capping around 70 before increasing again when adding additional features. The decision was then made to switch to a random forest model using three-fold cross validation to improve prediction accuracy. Doing so was at the cost of model interpretability. However, given the objective of this project was to generate predictions with the lowest RMSE, this compromise was deemed appropriate.

Additional feature-engineered variables were then added to the random forest model. The process of adding features into the random forest model is described in below.

Inference

After each model run, the test RMSE was calculated and compared with the proceeding model run. If the RMSE improved, then the additional features included were kept in the final model. If the RMSE increased, the model was first tuned by changing tuning the following parameters:

- the number of bootstraps used (ntree)
- the number of predictors used in each random forest (mtry)
- the number folds used for cross validation.

If tuning these parameters also resulted in a worsening of the RMSE, then these additional features were dropped and a different set of features were considered.

The final random forest model consisted of:

- 69 predictors
- 30 predictors used per tree
- three-fold cross validation
- 500 bootstraps (trees).

4 Final remarks

The main takeaway was that having a consistent project workflow greatly improved code readability, debugging and review throughout the project. By separating cleaning and modeling code into different files, it was easier to monitor, add and remove code to add additional features and test various models.

However, a number of improvements could potentially improve the RMSE score. Conducting a more thorough analysis of all text data, especially on those features related to the description of the property and the host, could have resulted in extracting more useful features to use for modeling. Additional knowledge of natural language processing techniques and methods could have been useful for constructing these additional features to improve RMSE.

Appendix A: A_Cleaning.R

```
library(tidyverse)
library(leaps)
library(glmnet)
library(gridExtra)
library(caret)
library(broom)
library(lubridate)
```

```r
library(zipcode)

########## PREAMBLE ##########

# Import the data and create and train and test set
mydata <- read.csv("Data/analysisData.csv", stringsAsFactors = F)
scoring <- read.csv("Data/scoringData.csv", stringsAsFactors = F)
data("zipcode")

set.seed(1251)
split <- createDataPartition(mydata$price, p = 0.7, list = F, groups = 50)
train <- mydata[split, ]
test <- mydata[-split, ]

########## ABOUT THE HOST ##########

clean_train <- train

# Feature engineering on details of the host
clean_train <- train %>%
  mutate(host_since2 = as.Date(host_since) - as.Date("2019-01-01")) %>%
  mutate(host_superhost = ifelse(host_is_superhost == "t", 1, 0),
         host_verified = ifelse(host_identity_verified == "t", 1, 0),
         host_about_length = nchar(host_about),
         host_response100 = ifelse(host_response_rate == "100%", 1, 0) %>% as.factor())

test <- test %>%
  mutate(host_since2 = as.Date(host_since) - as.Date("2019-01-01")) %>%
  mutate(host_superhost = ifelse(host_is_superhost == "t", 1, 0),
         host_verified = ifelse(host_identity_verified == "t", 1, 0),
         host_about_length = nchar(host_about),
         host_response100 = ifelse(host_response_rate == "100%", 1, 0) %>% as.factor())


scoring <- scoring %>%
  mutate(host_since2 = as.Date(host_since) - as.Date("2019-01-01")) %>%
  mutate(host_superhost = ifelse(host_is_superhost == "t", 1, 0),
         host_verified = ifelse(host_identity_verified == "t", 1, 0),
         host_about_length = nchar(host_about),
         host_response100 = ifelse(host_response_rate == "100%", 1, 0) %>% as.factor())

recent <- c("yesterday", "today", "2 days ago", "3 days ago", "4 days ago",
            "5 days ago", "6 days ago", "a week ago", "1 week ago", "2 weeks ago")

clean_train$recent_update <- ifelse(clean_train$calendar_updated %in% recent,
                                    "Recent", "Not recent") %>% as.factor()
test$recent_update <- ifelse(test$calendar_updated %in% recent,
                                    "Recent", "Not recent") %>% as.factor()
scoring$recent_update <- ifelse(scoring$calendar_updated %in% recent,
                                    "Recent", "Not recent") %>% as.factor()


########## LOCATION ##########
```

```r
### Latitude and longitude of zipcodes to use as additional features

clean_train <- left_join(clean_train, zipcode %>%
                           select(-city, -state), by = c("zipcode" = "zip"))
test <- left_join(test, zipcode %>%
                    select(-city, -state), by = c("zipcode" = "zip"))
scoring <- left_join(scoring, zipcode %>%
                       select(-city, -state), by = c("zipcode" = "zip"))

# What to do with missing lats and lons
clean_train$latitude[is.na(clean_train$latitude)] <-
  mean(clean_train$latitude, na.rm = T)
clean_train$longitude[is.na(clean_train$longitude)] <-
  mean(clean_train$longitude, na.rm = T)

test$latitude[is.na(test$latitude)] <-
  mean(test$latitude, na.rm = T)
test$longitude[is.na(test$longitude)] <-
  mean(test$longitude, na.rm = T)

scoring$latitude[is.na(scoring$latitude)] <-
  mean(scoring$latitude, na.rm = T)
scoring$longitude[is.na(scoring$longitude)] <-
  mean(scoring$longitude, na.rm = T)


########## PROPERTY ##########

# Feature engineering about the property
clean_train <- clean_train %>%
  mutate(name_length = nchar(name),
         space_length = nchar(space),
         description_length = nchar(description),
         has_notes = nchar(notes),
         has_transit = ifelse(nchar(transit) != 0, 1, 0),
         has_hrules = ifelse(nchar(house_rules) != 0, 1, 0),
         has_sdeposit = ifelse(security_deposit > 0, 1, 0),
         has_cleaning_fee = ifelse(cleaning_fee > 0, 1, 0),
         is_apartment = ifelse(property_type == "Apartment", 1, 0) %>% as.factor(),
         is_hotel = ifelse(property_type == "Hotel", 1, 0) %>% as.factor(),
         is_house = ifelse(property_type == "House", 1, 0) %>% as.factor(),
         is_loft = ifelse(property_type == "Loft", 1, 0) %>% as.factor(),
         is_townhouse = ifelse(property_type == "Townhouse", 1, 0) %>% as.factor(),
         is_condo = ifelse(property_type == "Condominium", 1, 0) %>% as.factor(),
         has_cleaning = ifelse(cleaning_fee > 0, 1, 0) %>% as.factor()) %>%
  filter(!is.na(beds))



test <- test %>%
  mutate(name_length = nchar(name),
         space_length = nchar(space),
         description_length = nchar(description),
```

```
         has_notes = nchar(notes),
         has_transit = ifelse(nchar(transit) != 0, 1, 0),
         has_hrules = ifelse(nchar(house_rules) != 0, 1, 0),
         has_sdeposit = ifelse(security_deposit > 0, 1, 0),
         has_cleaning_fee = ifelse(cleaning_fee > 0, 1, 0),
         is_apartment = ifelse(property_type == "Apartment", 1, 0) %>% as.factor(),
         is_hotel = ifelse(property_type == "Hotel", 1, 0) %>% as.factor(),
         is_house = ifelse(property_type == "House", 1, 0) %>% as.factor(),
         is_loft = ifelse(property_type == "Loft", 1, 0) %>% as.factor(),
         is_townhouse = ifelse(property_type == "Townhouse", 1, 0) %>% as.factor(),
         is_condo = ifelse(property_type == "Condominium", 1, 0) %>% as.factor(),
         has_cleaning = ifelse(cleaning_fee > 0, 1, 0) %>% as.factor()) %>%
  filter(!is.na(beds))

scoring <- scoring %>%
  mutate(name_length = nchar(name),
         space_length = nchar(space),
         description_length = nchar(description),
         has_notes = nchar(notes),
         has_transit = ifelse(nchar(transit) != 0, 1, 0),
         has_hrules = ifelse(nchar(house_rules) != 0, 1, 0),
         has_sdeposit = ifelse(security_deposit > 0, 1, 0),
         has_cleaning_fee = ifelse(cleaning_fee > 0, 1, 0),
         is_apartment = ifelse(property_type == "Apartment", 1, 0) %>% as.factor(),
         is_hotel = ifelse(property_type == "Hotel", 1, 0) %>% as.factor(),
         is_house = ifelse(property_type == "House", 1, 0) %>% as.factor(),
         is_loft = ifelse(property_type == "Loft", 1, 0) %>% as.factor(),
         is_townhouse = ifelse(property_type == "Townhouse", 1, 0) %>% as.factor(),
         is_condo = ifelse(property_type == "Condominium", 1, 0) %>% as.factor(),
         has_cleaning = ifelse(cleaning_fee > 0, 1, 0) %>% as.factor())


### Feature engineering from amenities
# TV
locs <- grepl(pattern = "TV", clean_train$amenities)
clean_train$has_TV <- 0
clean_train$has_TV[locs] <- 1
clean_train$has_TV <- as.factor(clean_train$has_TV)

# Wifi
locs <- grepl(pattern = "Wifi", clean_train$amenities)
clean_train$has_WF <- 0
clean_train$has_WF[locs] <- 1
clean_train$has_WF <- as.factor(clean_train$has_WF)

# Internet
locs <- grepl(pattern = "Air conditioning", clean_train$amenities)
clean_train$has_AC <- 0
clean_train$has_AC[locs] <- 1
clean_train$has_AC <- as.factor(clean_train$has_AC)

# Kitchen
locs <- grepl(pattern = "Kitchen", clean_train$amenities)
```

```
clean_train$has_KN <- 0
clean_train$has_KN[locs] <- 1
clean_train$has_KN <- as.factor(clean_train$has_KN)

# Internet
locs <- grepl(pattern = "Internet", clean_train$amenities)
clean_train$has_IT <- 0
clean_train$has_IT[locs] <- 1
clean_train$has_IT <- as.factor(clean_train$has_IT)

# Washer
locs <- grepl(pattern = "Washer", clean_train$amenities)
clean_train$has_WS <- 0
clean_train$has_WS[locs] <- 1
clean_train$has_WS <- as.factor(clean_train$has_WS)

# Heating
locs <- grepl(pattern = "Heating", clean_train$amenities)
clean_train$has_HT <- 0
clean_train$has_HT[locs] <- 1
clean_train$has_HT <- as.factor(clean_train$has_HT)



# TV
locs <- grepl(pattern = "TV", test$amenities)
test$has_TV <- 0
test$has_TV[locs] <- 1
test$has_TV <- as.factor(test$has_TV)

# Wifi
locs <- grepl(pattern = "Wifi", test$amenities)
test$has_WF <- 0
test$has_WF[locs] <- 1
test$has_WF <- as.factor(test$has_WF)

# Internet
locs <- grepl(pattern = "Air conditioning", test$amenities)
test$has_AC <- 0
test$has_AC[locs] <- 1
test$has_AC <- as.factor(test$has_AC)

# Kitchen
locs <- grepl(pattern = "Kitchen", test$amenities)
test$has_KN <- 0
test$has_KN[locs] <- 1
test$has_KN <- as.factor(test$has_KN)

# Internet
locs <- grepl(pattern = "Internet", test$amenities)
test$has_IT <- 0
test$has_IT[locs] <- 1
test$has_IT <- as.factor(test$has_IT)
```

```r
# Washer
locs <- grepl(pattern = "Washer", test$amenities)
test$has_WS <- 0
test$has_WS[locs] <- 1
test$has_WS <- as.factor(test$has_WS)

# Heating
locs <- grepl(pattern = "Heating", test$amenities)
test$has_HT <- 0
test$has_HT[locs] <- 1
test$has_HT <- as.factor(test$has_HT)


# TV
locs <- grepl(pattern = "TV", scoring$amenities)
scoring$has_TV <- 0
scoring$has_TV[locs] <- 1
scoring$has_TV <- as.factor(scoring$has_TV)

# Wifi
locs <- grepl(pattern = "Wifi", scoring$amenities)
scoring$has_WF <- 0
scoring$has_WF[locs] <- 1
scoring$has_WF <- as.factor(scoring$has_WF)

# Internet
locs <- grepl(pattern = "Air conditioning", scoring$amenities)
scoring$has_AC <- 0
scoring$has_AC[locs] <- 1
scoring$has_AC <- as.factor(scoring$has_AC)

# Kitchen
locs <- grepl(pattern = "Kitchen", scoring$amenities)
scoring$has_KN <- 0
scoring$has_KN[locs] <- 1
scoring$has_KN <- as.factor(scoring$has_KN)

# Internet
locs <- grepl(pattern = "Internet", scoring$amenities)
scoring$has_IT <- 0
scoring$has_IT[locs] <- 1
scoring$has_IT <- as.factor(scoring$has_IT)

# Washer
locs <- grepl(pattern = "Washer", scoring$amenities)
scoring$has_WS <- 0
scoring$has_WS[locs] <- 1
scoring$has_WS <- as.factor(scoring$has_WS)

# Heating
locs <- grepl(pattern = "Heating", scoring$amenities)
scoring$has_HT <- 0
scoring$has_HT[locs] <- 1
```

```
scoring$has_HT <- as.factor(scoring$has_HT)


# Breakfast

locs <- grepl(pattern = "Breakfast", clean_train$amenities)
clean_train$has_BF <- 0
clean_train$has_BF[locs] <- 1
clean_train$has_BF <- as.factor(clean_train$has_BF)

locs <- grepl(pattern = "Breakfast", test$amenities)
test$has_BF <- 0
test$has_BF[locs] <- 1
test$has_BF <- as.factor(test$has_BF)

locs <- grepl(pattern = "Breakfast", scoring$amenities)
scoring$has_BF <- 0
scoring$has_BF[locs] <- 1
scoring$has_BF <- as.factor(scoring$has_BF)


# Gym
locs <- grepl(pattern = "Gym", clean_train$amenities)
clean_train$has_GY <- 0
clean_train$has_GY[locs] <- 1
clean_train$has_GY <- as.factor(clean_train$has_GY)

locs <- grepl(pattern = "Gym", test$amenities)
test$has_GY <- 0
test$has_GY[locs] <- 1
test$has_GY <- as.factor(test$has_GY)

locs <- grepl(pattern = "Gym", scoring$amenities)
scoring$has_GY <- 0
scoring$has_GY[locs] <- 1
scoring$has_GY <- as.factor(scoring$has_GY)


locs <- grepl(pattern = "Doorman", clean_train$amenities)
clean_train$has_DM <- 0
clean_train$has_DM[locs] <- 1
clean_train$has_DM <- as.factor(clean_train$has_DM)
locs <- grepl(pattern = "Doorman", test$amenities)
test$has_DM <- 0
test$has_DM[locs] <- 1
test$has_DM <- as.factor(test$has_DM)
locs <- grepl(pattern = "Doorman", scoring$amenities)
scoring$has_DM <- 0
scoring$has_DM[locs] <- 1
scoring$has_DM <- as.factor(scoring$has_DM)

locs <- grepl(pattern = "Elevator", clean_train$amenities)
clean_train$has_EV <- 0
```

```
clean_train$has_EV[locs] <- 1
clean_train$has_EV <- as.factor(clean_train$has_EV)
locs <- grepl(pattern = "Elevator", test$amenities)
test$has_EV <- 0
test$has_EV[locs] <- 1
test$has_EV <- as.factor(test$has_EV)
locs <- grepl(pattern = "Elevator", scoring$amenities)
scoring$has_EV <- 0
scoring$has_EV[locs] <- 1
scoring$has_EV <- as.factor(scoring$has_EV)

locs <- grepl(pattern = "Pool", clean_train$amenities)
clean_train$has_PL <- 0
clean_train$has_PL[locs] <- 1
clean_train$has_PL <- as.factor(clean_train$has_PL)
locs <- grepl(pattern = "Pool", test$amenities)
test$has_PL <- 0
test$has_PL[locs] <- 1
test$has_PL <- as.factor(test$has_PL)
locs <- grepl(pattern = "Pool", scoring$amenities)
scoring$has_PL <- 0
scoring$has_PL[locs] <- 1
scoring$has_PL <- as.factor(scoring$has_PL)

locs <- grepl(pattern = "Essentials", clean_train$amenities)
clean_train$has_ES <- 0
clean_train$has_ES[locs] <- 1
clean_train$has_ES <- as.factor(clean_train$has_ES)
locs <- grepl(pattern = "Essentials", test$amenities)
test$has_ES <- 0
test$has_ES[locs] <- 1
test$has_ES <- as.factor(test$has_ES)
locs <- grepl(pattern = "Essentials", scoring$amenities)
scoring$has_ES <- 0
scoring$has_ES[locs] <- 1
scoring$has_ES <- as.factor(scoring$has_ES)

# Has subway
locs <- grepl(pattern = "subway", clean_train$description)
clean_train$has_SB <- 0
clean_train$has_SB[locs] <- 1
clean_train$has_SB <- as.factor(clean_train$has_SB)

locs <- grepl(pattern = "subway", test$description)
test$has_SB <- 0
test$has_SB[locs] <- 1
test$has_SB <- as.factor(test$has_SB)

locs <- grepl(pattern = "subway", scoring$description)
scoring$has_SB <- 0
scoring$has_SB[locs] <- 1
scoring$has_SB <- as.factor(scoring$has_SB)
```

```
########## REVIEWS ##########

clean_train <- clean_train %>%
  mutate(first_review = as.Date(first_review),
         last_review = as.Date(last_review)) %>%
  mutate(first_rev_days = as.Date("2019-01-01") - first_review,
         last_rev_days = as.Date("2019-01-01") - last_review)

test <- test %>%
  mutate(first_review = as.Date(first_review),
         last_review = as.Date(last_review)) %>%
  mutate(first_rev_days = as.Date("2019-01-01") - first_review,
         last_rev_days = as.Date("2019-01-01") - last_review)

scoring <- scoring %>%
  mutate(first_review = as.Date(first_review),
         last_review = as.Date(last_review)) %>%
  mutate(first_rev_days = as.Date("2019-01-01") - first_review,
         last_rev_days = as.Date("2019-01-01") - last_review)


### Listing time = Last review - first review
clean_train$first_review <- as.Date(clean_train$first_review)
clean_train$last_review <- as.Date(clean_train$last_review)
clean_train$listing_time <- clean_train$last_review - clean_train$first_review

test$first_review <- as.Date(test$first_review)
test$last_review <- as.Date(test$last_review)
test$listing_time <- test$last_review - test$first_review

scoring$first_review <- as.Date(scoring$first_review)
scoring$last_review <- as.Date(scoring$last_review)
scoring$listing_time <- scoring$last_review - scoring$first_review


### More cleaning - repeated on clean, test and scoring

clean_train$host_since2[is.na(clean_train$host_since2)] <-
  mean(clean_train$host_since2, na.rm = T)
clean_train$first_rev_days[is.na(clean_train$first_rev_days)] <-
  mean(clean_train$first_rev_days, na.rm = T)
clean_train$last_rev_days[is.na(clean_train$last_rev_days)] <-
  mean(clean_train$last_rev_days, na.rm = T)
clean_train$reviews_per_month[is.na(clean_train$reviews_per_month)] <-
  mean(clean_train$reviews_per_month, na.rm = T)
clean_train$has_sdeposit[is.na(clean_train$has_sdeposit)] <- 0
clean_train$is_apartment[is.na(clean_train$is_aparment)] <- 0
clean_train$has_cleaning[is.na(clean_train$has_cleaning)] <- 0
clean_train$listing_time[is.na(clean_train$listing_time)] <-
  mean(clean_train$listing_time, na.rm = T)
clean_train$cleaning_fee[is.na(clean_train$cleaning_fee)] <-
  mean(clean_train$cleaning_fee, na.rm = T)
clean_train$host_since <- as.Date(clean_train$host_since)
```

```
clean_train$host_year <- year(clean_train$host_since)
clean_train$host_year[is.na(clean_train$host_year)] <-
  mean(clean_train$host_year, na.rm = T)
clean_train$host_about_length[is.na(clean_train$host_about_length)] <-
  mean(clean_train$host_about_length, na.rm = T)

test$host_since2[is.na(test$host_since2)] <-
  mean(test$host_since2, na.rm = T)
test$first_rev_days[is.na(test$first_rev_days)] <-
  mean(test$first_rev_days, na.rm = T)
test$last_rev_days[is.na(test$last_rev_days)] <-
  mean(test$last_rev_days, na.rm = T)
test$reviews_per_month[is.na(test$reviews_per_month)] <-
  mean(test$reviews_per_month, na.rm = T)
test$has_sdeposit[is.na(test$has_sdeposit)] <- 0
test$is_apartment[is.na(test$is_aparment)] <- 0
test$has_cleaning[is.na(test$has_cleaning)] <- 0
test$listing_time[is.na(test$listing_time)] <-
  mean(test$listing_time, na.rm = T)
test$cleaning_fee[is.na(test$cleaning_fee)] <-
  mean(test$cleaning_fee, na.rm = T)
test$host_since <- as.Date(test$host_since)
test$host_year <- year(test$host_since)
test$host_year[is.na(test$host_year)] <-
  mean(test$host_year, na.rm = T)
test$host_about_length[is.na(test$host_about_length)] <-
  mean(test$host_about_length, na.rm = T)


scoring$host_since2[is.na(scoring$host_since2)] <-
  mean(scoring$host_since2, na.rm = T)
scoring$first_rev_days[is.na(scoring$first_rev_days)] <-
  mean(scoring$first_rev_days, na.rm = T)
scoring$last_rev_days[is.na(scoring$last_rev_days)] <-
  mean(scoring$last_rev_days, na.rm = T)
scoring$reviews_per_month[is.na(scoring$reviews_per_month)] <-
  mean(scoring$reviews_per_month, na.rm = T)
scoring$has_sdeposit[is.na(scoring$has_sdeposit)] <- 0
scoring$is_apartment[is.na(scoring$is_aparment)] <- 0
scoring$has_cleaning[is.na(scoring$has_cleaning)] <- 0
scoring$listing_time[is.na(scoring$listing_time)] <-
  mean(scoring$listing_time, na.rm = T)
scoring$cleaning_fee[is.na(scoring$cleaning_fee)] <-
  mean(scoring$cleaning_fee, na.rm = T)
scoring$host_since <- as.Date(scoring$host_since)
scoring$host_year <- year(scoring$host_since)
scoring$host_year[is.na(scoring$host_year)] <-
  mean(scoring$host_year, na.rm = T)
scoring$host_about_length[is.na(scoring$host_about_length)] <-
  mean(scoring$host_about_length, na.rm = T)


########## SAVE CLEANING OUTPUTS ##########
```

```r
save(clean_train, file = "Cleaned_data/train.Rdata")
save(test, file = "Cleaned_data/test.Rdata")
save(scoring, file = "Cleaned_data/scoring.Rdata")
```

Appendix B: B_MODELING.R

```r
library(tidyverse)
library(gridExtra)
library(caret)
library(randomForest)

rm(list = ls())

load("Cleaned_data/train.Rdata")
load("Cleaned_data/test.Rdata")
load("Cleaned_data/scoring.Rdata")

##### Turn variables to factors
f_vars <- c("host_superhost", "host_verified",
            "neighbourhood_group_cleansed", "is_location_exact",
            "has_transit", "has_hrules", "property_type", "room_type",
            "bed_type", "has_sdeposit", "instant_bookable", "cancellation_policy")

clean_train[f_vars] <- lapply(clean_train[f_vars], factor)
test[f_vars] <- lapply(test[f_vars], factor)
scoring[f_vars] <- lapply(scoring[f_vars], factor)

clean_train$square_feet2 <- ifelse(is.na(clean_train$square_feet),
                                   mean(clean_train$square_feet, na.rm = T),
                                   clean_train$square_feet)

test$square_feet2 <- ifelse(is.na(test$square_feet),
                                   mean(test$square_feet, na.rm = T),
                                   test$square_feet)

scoring$square_feet2 <- ifelse(is.na(scoring$square_feet),
                                   mean(scoring$square_feet, na.rm = T),
                                   scoring$square_feet)



########## THE MODEL ##########

# Model being used for modeling
model_formula <- price ~
  host_superhost + host_verified + host_since2 +
  calculated_host_listings_count_entire_homes +
  calculated_host_listings_count_private_rooms +
  calculated_host_listings_count_shared_rooms +
  neighbourhood_group_cleansed + is_location_exact +
  number_of_reviews_ltm + bed_type + instant_bookable +
  cancellation_policy + description_length +
```

```
  name_length + space_length + bathrooms +
  bedrooms + beds + latitude + longitude +
  minimum_nights_avg_ntm + availability_365 +
  review_scores_value + last_rev_days +
  reviews_per_month + room_type + guests_included +
  extra_people + accommodates + availability_30 +
  review_scores_rating +
  review_scores_cleanliness +
  review_scores_location +
  has_transit + has_hrules +
  has_sdeposit +
  availability_60 + review_scores_checkin +
  review_scores_communication + review_scores_accuracy +
  has_IT + has_WF + has_AC + has_KN + has_WS + has_HT +
  has_TV + has_BF + has_GY + has_SB + availability_90 +
  has_DM + has_EV + has_PL + has_ES +
  is_apartment + has_cleaning + listing_time +
  cleaning_fee + host_year +
  is_hotel + is_condo + is_house + is_loft +
  is_townhouse +
  require_guest_phone_verification + host_about_length +
  host_response100 + recent_update


########## RANDOM FOREST CROSS VALIDATION ##########

# CROSS VALIDATION TO FIND OPTIMAL MTRY
set.seed(258)
# CV random forest - remove coefficients that don't make sense
trControl <- trainControl(method= "cv", number = 3)
tuneGrid <- expand.grid(mtry = 30)

cvForest <- train(model_formula,
                  clean_train,
                  method = "rf",
                  ntree = 500,
                  trControl = trControl,
                  tuneGrid = tuneGrid)

cvForest
pred <- predict(cvForest, newdata = test)
rmse <- mean((pred - test$price)^2) %>% sqrt()
rmse

########## SCORE THE MODEL ##########

scoring$host_identity_verified[which(scoring$host_identity_verified == "")] <- "f"
scoring$host_identity_verified <- factor(scoring$host_identity_verified, levels = c("t", "f"))

# Impute missing values in scoring data with means
score.missing <- function(scoring){
  lapply(scoring, function(x){sum(is.na(x))}) %>% unlist()
}
```

```r
scoring$host_listings_count[which(is.na(scoring$host_listings_count))] <-
  mean(scoring$host_listings_count, na.rm = T)
scoring$beds[which(is.na(scoring$beds))] <-
  mean(scoring$beds, na.rm = T)


pred_s <- predict(cvForest, newdata = scoring)
submission <- data.frame(id = scoring$id, price = pred_s)
# submission$price[which(is.na(submission$price))] <- mean(submission$price, na.rm = T)

write_csv(submission, "submission.csv")
```