Classification Part 3

2023-08-13

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.0     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
ts to become errors
```

```
setwd('/Users/salman/Desktop/')
df <- read_csv('hong_kong_cleaned.csv')
```

```
## New names:
## Rows: 1424 Columns: 50
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," chr
## (10): description, neighborhood_overview, host_location, host_response_... dbl
## (34): ...1, host_id, host_response_rate, host_acceptance_rate, host_lis... lgl
## (5): host_is_superhost, host_has_profile_pic, host_identity_verified, ... date
## (1): host_since
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## • `` -> `...1`
```

Before building the model let us create a training and data set.

```
set.seed(979)
indexes <-sample(seq(1,nrow(df)), 0.6*nrow(df))
training <- df[indexes,]
validation <- df[-indexes,]
```

We have to predict instant booking which lets customers book an aprtment without sending a request to the host first. This means there is some level of trust and convenience that is established.

We have to choose the variables to be used in the model. The decision tree models already select which feature are most relevant so we don't have to work too much here. First some variables have to be converted from text to categories for the model to be able to use them.

```
cols <- c("host_response_time", "host_neighbourhood", "host_verifications",
          "room_type", "bathroom_type", "property_type")

df[cols] <- lapply(df[cols], factor)
```

The variables we choose for the model are the following :

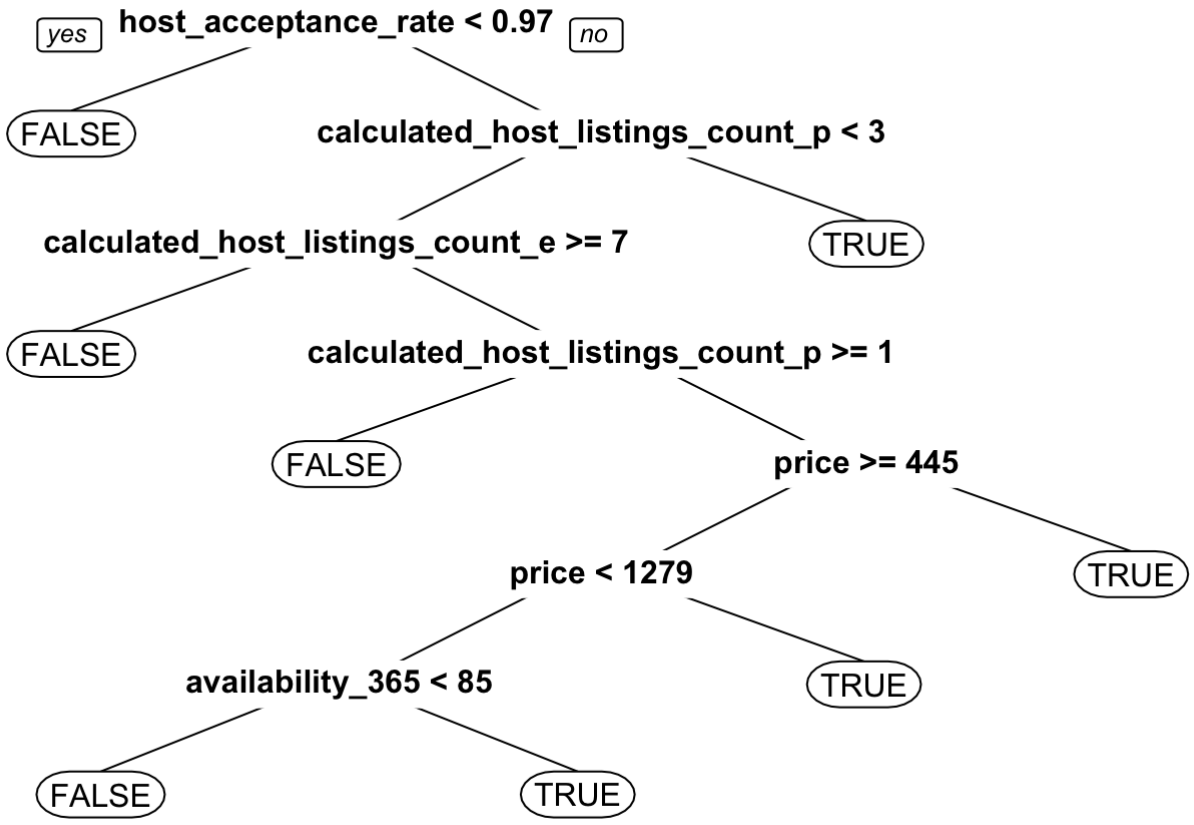
host related variables as they could indicate the reliability of the host.

‘room_type’, ‘beds’, ‘price’ contains information on the characteristics of the property ‘availability_30’,‘availability_60’,‘availability_90’ can indicate listing demand or popularity. ‘got_reviewed’ and review scores show reputation and guest experience. host_listings count also show experience

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

```
cl_tree  <- rpart(instant_bookable ~
                    host_response_time + host_response_rate + host_acceptance_rate +
host_is_superhost +  host_has_profile_pic + host_identity_verified +

                  room_type + bathroom_nb + beds + price +

                  availability_30 + availability_60 + availability_90 +
                  availability_365 +

                  number_of_reviews +
                  calculated_host_listings_count_entire_homes +
                  calculated_host_listings_count_private_rooms +
                  calculated_host_listings_count_shared_rooms,

                data = training,
                method = "class")
```

```
prp(cl_tree)
```




```
library(caret)
```

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

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

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

```
pred_t <- predict(cl_tree, training, type="class")
pred_v <- predict(cl_tree, validation, type="class")

confusion_t <- table(pred_t, training$instant_bookable)

confusionMatrix(confusion_t)
```

```
## Confusion Matrix and Statistics
##
##
## pred_t  FALSE TRUE
##   FALSE   698   51
##   TRUE     16   89
##
##                Accuracy : 0.9215
##                  95% CI : (0.9014, 0.9387)
##     No Information Rate : 0.8361
##     P-Value [Acc > NIR] : 1.318e-13
##
##                   Kappa : 0.6818
##
##  Mcnemar's Test P-Value : 3.271e-05
##
##             Sensitivity : 0.9776
##             Specificity : 0.6357
##          Pos Pred Value : 0.9319
##          Neg Pred Value : 0.8476
##              Prevalence : 0.8361
##          Detection Rate : 0.8173
##    Detection Prevalence : 0.8770
##       Balanced Accuracy : 0.8067
##
##        'Positive' Class : FALSE
##
```

```
confusion_v <- table(pred_v, validation$instant_bookable)

confusionMatrix(confusion_v)
```

```
## Confusion Matrix and Statistics
##
##
## pred_v  FALSE TRUE
##   FALSE   449   42
##   TRUE     18   61
##
##                Accuracy : 0.8947
##                  95% CI : (0.8666, 0.9187)
##     No Information Rate : 0.8193
##     P-Value [Acc > NIR] : 4.246e-07
##
##                   Kappa : 0.609
##
##  Mcnemar's Test P-Value : 0.002985
##
##             Sensitivity : 0.9615
##             Specificity : 0.5922
##          Pos Pred Value : 0.9145
##          Neg Pred Value : 0.7722
##              Prevalence : 0.8193
##          Detection Rate : 0.7877
##    Detection Prevalence : 0.8614
##       Balanced Accuracy : 0.7768
##
##        'Positive' Class : FALSE
##
```

The model has high accuracy of 87% without optimizing the complexity parameter. Overfitting is already low by comparing with the accuracy of 91% of the training set.

B.

Now let try and find the optimal complexity parameter.

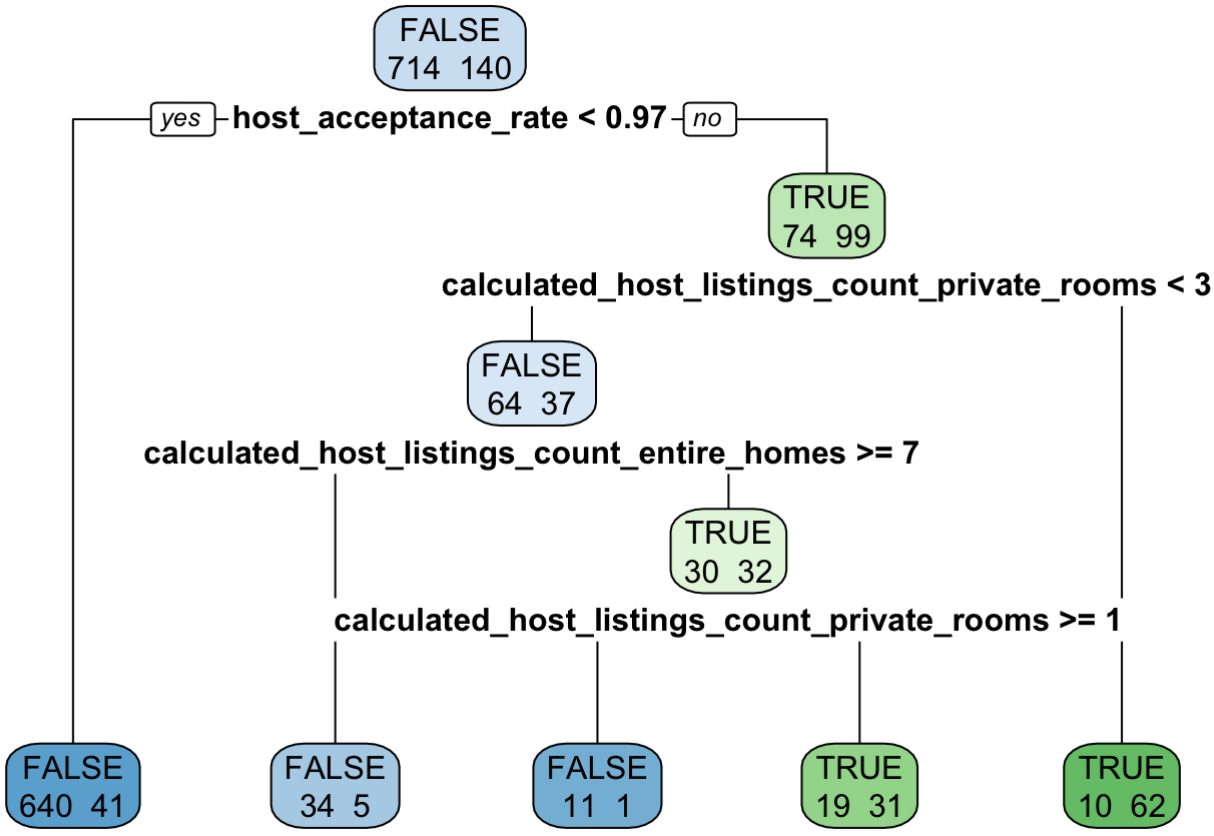
```
cv.ct <- rpart(instant_bookable ~
                    host_response_time + host_response_rate + host_acceptance_rate +
host_is_superhost + host_verifications + host_has_profile_pic + host_identity_verifie
d +

                  room_type + bathroom_nb + bathroom_type + beds + price +

                  availability_30 + availability_60 + availability_90 +
                  availability_365 +

                  number_of_reviews +
                  calculated_host_listings_count_entire_homes +
                  calculated_host_listings_count_private_rooms +
                  calculated_host_listings_count_shared_rooms,
               data = training,
               method = "class", minsplit = 5 , cp=0.00001, xval = 10)

printcp(cv.ct)
```

```
##
## Classification tree:
## rpart(formula = instant_bookable ~ host_response_time + host_response_rate +
##     host_acceptance_rate + host_is_superhost + host_verifications +
##     host_has_profile_pic + host_identity_verified + room_type +
##     bathroom_nb + bathroom_type + beds + price + availability_30 +
##     availability_60 + availability_90 + availability_365 + number_of_reviews +
##     calculated_host_listings_count_entire_homes + calculated_host_listings_count_p
rivate_rooms +
##     calculated_host_listings_count_shared_rooms, data = training,
##     method = "class", minsplit = 5, cp = 1e-05, xval = 10)
##
## Variables actually used in tree construction:
##  [1] availability_30
##  [2] availability_365
##  [3] availability_60
##  [4] availability_90
##  [5] bathroom_nb
##  [6] beds
##  [7] calculated_host_listings_count_entire_homes
##  [8] calculated_host_listings_count_private_rooms
##  [9] host_acceptance_rate
## [10] host_identity_verified
## [11] host_response_rate
## [12] host_response_time
## [13] host_verifications
## [14] number_of_reviews
## [15] price
##
## Root node error: 140/854 = 0.16393
##
## n= 854
##
##           CP nsplit rel error  xerror     xstd
## 1  0.1857143      0   1.00000 1.00000 0.077278
## 2  0.0428571      2   0.62857 0.70714 0.066824
## 3  0.0357143      4   0.54286 0.67857 0.065634
## 4  0.0285714      5   0.50714 0.67857 0.065634
## 5  0.0214286      6   0.47857 0.62143 0.063139
## 6  0.0142857      8   0.43571 0.57143 0.060822
## 7  0.0107143     18   0.28571 0.59286 0.061831
## 8  0.0071429     20   0.26429 0.58571 0.061498
## 9  0.0053571     25   0.22857 0.58571 0.061498
## 10 0.0035714     29   0.20714 0.60714 0.062491
## 11 0.0000100     38   0.17143 0.65714 0.064716
```

The optimal complexity parameter to reduce error is 0.0223881

C.

```
cl_tree  <- rpart(instant_bookable ~
                    host_response_time + host_response_rate + host_acceptance_rate +
host_is_superhost + host_has_profile_pic + host_identity_verified +

                  room_type + bathroom_nb + beds + price +

                  availability_30 + availability_60 + availability_90 +
                  availability_365 +

                  number_of_reviews +
                  calculated_host_listings_count_entire_homes +
                  calculated_host_listings_count_private_rooms +
                  calculated_host_listings_count_shared_rooms,

                data = training,
                method = "class", cp = 0.0223881)

rpart.plot(cl_tree, type =2, extra =1)
```



Let us compare the confusion matrix on both the training and testing dataset to evaluate the model.

```
pred_t <- predict(cl_tree, training, type="class")
pred_v <- predict(cl_tree, validation, type="class")

confusion_t <- table(pred_t, training$instant_bookable)

confusionMatrix(confusion_t)
```

```
## Confusion Matrix and Statistics
##
##
## pred_t  FALSE TRUE
##   FALSE   685   47
##   TRUE     29   93
##
##                Accuracy : 0.911
##                  95% CI : (0.8899, 0.9292)
##     No Information Rate : 0.8361
##     P-Value [Acc > NIR] : 1.356e-10
##
##                   Kappa : 0.6577
##
##  Mcnemar's Test P-Value : 0.05117
##
##             Sensitivity : 0.9594
##             Specificity : 0.6643
##          Pos Pred Value : 0.9358
##          Neg Pred Value : 0.7623
##              Prevalence : 0.8361
##          Detection Rate : 0.8021
##    Detection Prevalence : 0.8571
##       Balanced Accuracy : 0.8118
##
##        'Positive' Class : FALSE
##
```

```
confusion_v <- table(pred_v, validation$instant_bookable)
confusionMatrix(confusion_v)
```

```
## Confusion Matrix and Statistics
##
##
## pred_v  FALSE TRUE
##   FALSE   437   40
##   TRUE     30   63
##
##                Accuracy : 0.8772
##                  95% CI : (0.8474, 0.903)
##     No Information Rate : 0.8193
##     P-Value [Acc > NIR] : 0.0001095
##
##                   Kappa : 0.5689
##
##  Mcnemar's Test P-Value : 0.2820589
##
##             Sensitivity : 0.9358
##             Specificity : 0.6117
##          Pos Pred Value : 0.9161
##          Neg Pred Value : 0.6774
##              Prevalence : 0.8193
##          Detection Rate : 0.7667
##    Detection Prevalence : 0.8368
##       Balanced Accuracy : 0.7737
##
##        'Positive' Class : FALSE
##
```

The model's accuracy on the training set is 91% while on the validation set it is 87.7% it is almost the same as the first tree. The first tree uses a combination of availability and number of reviews and addition to price and host acceptance rate. WHile the second tree uses almost exclusively the host listings count by room type after the root node. This second tree also has less depth compared to the first one meaning it can be generalized better. We can get the same accuracy through multiple ways of splitting but we would take the second tree as predictor before of its depth.

The sensitivy is very high at 0.93 while the specificity is lower at 0.61. Since the model put the positive class as FALSE, this means that the model is much better at identifying an airbnbn that is not instantly bookable.

One more model that could be tried would be to add property_type which we chose to remove because it had too many categories.

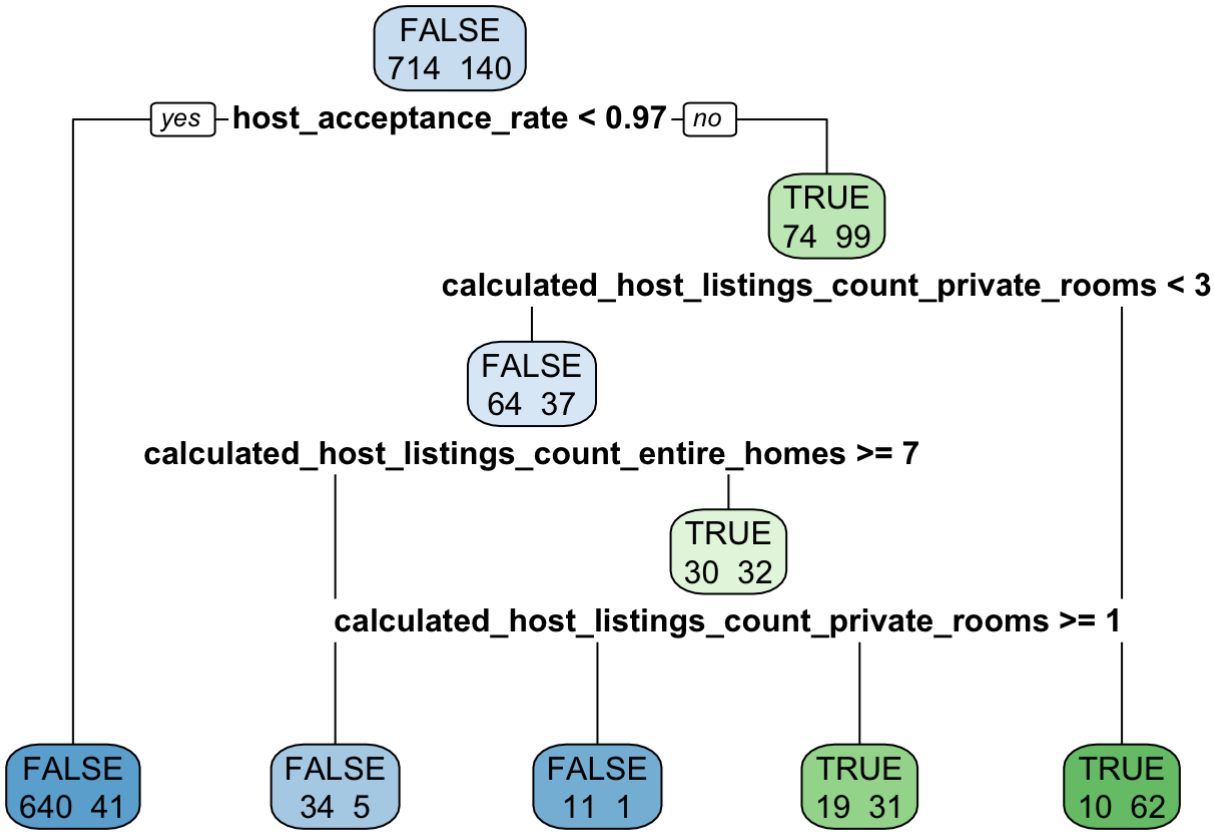
```
cv.ct <- rpart(instant_bookable ~
                    host_response_time + host_response_rate + host_acceptance_rate +
host_is_superhost + host_verifications + host_has_profile_pic + host_identity_verifie
d +

                  room_type + bathroom_nb + bathroom_type + beds + price +
                 property_type +

                  availability_30 + availability_60 + availability_90 +
                  availability_365 +

                  number_of_reviews +
                  calculated_host_listings_count_entire_homes +
                  calculated_host_listings_count_private_rooms +
                  calculated_host_listings_count_shared_rooms + got_reviewed,
               data = training,
               method = "class", minsplit = 5 , cp=0.00001, xval = 10)

printcp(cv.ct)
```

```
##
## Classification tree:
## rpart(formula = instant_bookable ~ host_response_time + host_response_rate +
##     host_acceptance_rate + host_is_superhost + host_verifications +
##     host_has_profile_pic + host_identity_verified + room_type +
##     bathroom_nb + bathroom_type + beds + price + property_type +
##     availability_30 + availability_60 + availability_90 + availability_365 +
##     number_of_reviews + calculated_host_listings_count_entire_homes +
##     calculated_host_listings_count_private_rooms + calculated_host_listings_count_
shared_rooms +
##     got_reviewed, data = training, method = "class", minsplit = 5,
##     cp = 1e-05, xval = 10)
##
## Variables actually used in tree construction:
##  [1] availability_30
##  [2] availability_365
##  [3] availability_60
##  [4] availability_90
##  [5] beds
##  [6] calculated_host_listings_count_entire_homes
##  [7] calculated_host_listings_count_private_rooms
##  [8] host_acceptance_rate
##  [9] host_identity_verified
## [10] host_response_rate
## [11] host_response_time
## [12] host_verifications
## [13] number_of_reviews
## [14] price
## [15] property_type
##
## Root node error: 140/854 = 0.16393
##
## n= 854
##
##           CP nsplit rel error  xerror     xstd
## 1  0.2035714      0   1.00000 1.00000 0.077278
## 2  0.0500000      2   0.59286 0.61429 0.062816
## 3  0.0285714      3   0.54286 0.61429 0.062816
## 4  0.0214286      4   0.51429 0.61429 0.062816
## 5  0.0178571      5   0.49286 0.60000 0.062162
## 6  0.0142857      9   0.42143 0.60000 0.062162
## 7  0.0071429     17   0.30000 0.59286 0.061831
## 8  0.0053571     29   0.21429 0.57857 0.061161
## 9  0.0023810     33   0.19286 0.60000 0.062162
## 10 0.0000100     41   0.15714 0.63571 0.063778
```

```
cv.ct <- rpart(instant_bookable ~
                    host_response_time + host_response_rate + host_acceptance_rate +
host_is_superhost + host_verifications + host_has_profile_pic + host_identity_verifie
d +

                  room_type + bathroom_nb + bathroom_type + beds + price +
                 property_type +

                  availability_30 + availability_60 + availability_90 +
                  availability_365 +

                  number_of_reviews +
                  calculated_host_listings_count_entire_homes +
                  calculated_host_listings_count_private_rooms +
                  calculated_host_listings_count_shared_rooms + got_reviewed,
               data = training,
               method = "class", cp=0.0071429)

pred_v <- predict(cl_tree, validation, type="class")
confusion_v <- table(pred_v, validation$instant_bookable)
confusionMatrix(confusion_v)
```

```
## Confusion Matrix and Statistics
##
##
## pred_v  FALSE TRUE
##   FALSE   437   40
##   TRUE     30   63
##
##                Accuracy : 0.8772
##                  95% CI : (0.8474, 0.903)
##     No Information Rate : 0.8193
##     P-Value [Acc > NIR] : 0.0001095
##
##                   Kappa : 0.5689
##
##  Mcnemar's Test P-Value : 0.2820589
##
##             Sensitivity : 0.9358
##             Specificity : 0.6117
##          Pos Pred Value : 0.9161
##          Neg Pred Value : 0.6774
##              Prevalence : 0.8193
##          Detection Rate : 0.7667
##    Detection Prevalence : 0.8368
##       Balanced Accuracy : 0.7737
##
##        'Positive' Class : FALSE
##
```

```
rpart.plot(cl_tree, type =2, extra =1)
```



The tree and accuracy are the same the variable did not have an impact.

D.

In this part we tried building a classifcation to predict if an apartment would be instantly bookable. To do that we first chose the most relevant variables for this prediction. Then we built a first classification tree and testes its accuracy and got 87% with little overfitting. For all trees the root node was host acceptance rate close to 10)% which is coherent with the fact that being booked instantly means an acceptance rate of 100%.

Then we tried optimizing the complexity parameter and got a new model with the same results on accuracy except with less depth and using very different splitting variables focusing on listing counts. Finally we tried adding property_type which did not have an impact on the tree. By looking at sensitivity and specificity we saw that the model is very good at predicting when a property will not be instantly bookable but it is much less accurate at predicting when it is instantly bookable. This stems from the fact that the host_acceptance rate already categorizes most FALSE accurately as we can see in the tree above.

Step IV: Clustering

In this part we did III. before II. since the information from the former is

used to define the latter.

I.

First let us take care of outliers we will remove the values above the 99% quantile :

```
library(tidyverse)
df2 <- filter(df, df$price < quantile(df$price, 0.99))
df2 <- filter(df2, df2$minimum_nights < quantile(df$minimum_nights, 0.99))
df2 <- filter(df2, df2$beds < quantile(df$beds, 0.99))
df2 <- filter(df2, df2$number_of_reviews < quantile(df$number_of_reviews, 0.99))
df2 <- filter(df2, df2$host_listings_count < quantile(df$host_listings_count, 0.99))
df2 <- filter(df2, df2$reviews_per_month < quantile(df$reviews_per_month, 0.99))
```

Clustering will use numerical variables we take the relevant ones for now.

```
numeric <- dplyr::select(df2, c('host_response_rate', 'host_acceptance_rate',
                                'host_listings_count', 'bathroom_nb',
                                'beds', 'price', 'minimum_nights',
                                'maximum_nights', 'availability_30',
                                'availability_60', 'availability_90',
                                'availability_365', 'number_of_reviews',
                                'review_scores_rating',
                                "calculated_host_listings_count_entire_homes",
                               "calculated_host_listings_count_private_rooms",
                               "calculated_host_listings_count_shared_rooms",
                                "reviews_per_month"))
```

We choose to use k-means clustering for this data as it will let us identify clusters more easily and find different type of properties in Wan Chai.

Now let us check the correlation between the variables. This will help us reduce the amount of variables for the model too many will make it hard to interprete.

```
corr <- cor(numeric)
colnames(corr) <- abbreviate(colnames(corr), 6)

M <- corr
print(round(M, 2))
```

```
##                                                hst_r_ hst_c_ hst_l_ bthrm_  beds
## host_response_rate                               1.00   0.23   0.14  -0.04 -0.03
## host_acceptance_rate                             0.23   1.00  -0.68   0.11  0.24
## host_listings_count                              0.14  -0.68   1.00  -0.08 -0.29
## bathroom_nb                                     -0.04   0.11  -0.08   1.00  0.31
## beds                                            -0.03   0.24  -0.29   0.31  1.00
## price                                           -0.21   0.34  -0.55   0.17  0.45
## minimum_nights                                   0.03  -0.37   0.49  -0.14 -0.27
## maximum_nights                                   0.06  -0.35   0.52   0.06 -0.03
## availability_30                                  0.06  -0.42   0.61   0.01 -0.16
## availability_60                                  0.08  -0.40   0.60   0.02 -0.17
## availability_90                                  0.08  -0.40   0.60   0.02 -0.15
## availability_365                                 0.10  -0.38   0.64   0.01 -0.12
## number_of_reviews                                0.03   0.24  -0.29   0.07  0.17
## review_scores_rating                            -0.02   0.45  -0.55   0.08  0.24
## calculated_host_listings_count_entire_homes      0.09  -0.31   0.35  -0.13 -0.08
## calculated_host_listings_count_private_rooms     0.14  -0.62   0.96  -0.06 -0.29
## calculated_host_listings_count_shared_rooms      0.16  -0.19   0.63   0.05 -0.12
## reviews_per_month                                0.01   0.36  -0.39   0.07  0.10
##                                                price mnmm_n mxmm_n avl_30 avl_60
## host_response_rate                            -0.21   0.03   0.06   0.06   0.08
## host_acceptance_rate                           0.34  -0.37  -0.35  -0.42  -0.40
## host_listings_count                           -0.55   0.49   0.52   0.61   0.60
## bathroom_nb                                    0.17  -0.14   0.06   0.01   0.02
## beds                                           0.45  -0.27  -0.03  -0.16  -0.17
## price                                          1.00  -0.35  -0.24  -0.38  -0.39
## minimum_nights                                -0.35   1.00   0.29   0.30   0.30
## maximum_nights                                -0.24   0.29   1.00   0.39   0.39
## availability_30                               -0.38   0.30   0.39   1.00   0.97
## availability_60                               -0.39   0.30   0.39   0.97   1.00
## availability_90                               -0.38   0.27   0.39   0.93   0.98
## availability_365                              -0.36   0.32   0.37   0.71   0.74
## number_of_reviews                              0.20  -0.24  -0.06  -0.24  -0.24
## review_scores_rating                           0.32  -0.43  -0.25  -0.40  -0.40
## calculated_host_listings_count_entire_homes   -0.07   0.32   0.30   0.28   0.28
## calculated_host_listings_count_private_rooms  -0.57   0.46   0.47   0.56   0.55
## calculated_host_listings_count_shared_rooms   -0.41   0.27   0.47   0.47   0.46
## reviews_per_month                              0.19  -0.40  -0.29  -0.28  -0.26
##                                                avl_90 av_365 nmbr__ rvw_s_
## host_response_rate                               0.08   0.10   0.03  -0.02
## host_acceptance_rate                            -0.40  -0.38   0.24   0.45
## host_listings_count                              0.60   0.64  -0.29  -0.55
## bathroom_nb                                      0.02   0.01   0.07   0.08
## beds                                            -0.15  -0.12   0.17   0.24
## price                                           -0.38  -0.36   0.20   0.32
## minimum_nights                                   0.27   0.32  -0.24  -0.43
## maximum_nights                                   0.39   0.37  -0.06  -0.25
## availability_30                                  0.93   0.71  -0.24  -0.40
## availability_60                                  0.98   0.74  -0.24  -0.40
## availability_90                                  1.00   0.79  -0.21  -0.37
## availability_365                                 0.79   1.00  -0.23  -0.40
## number_of_reviews                               -0.21  -0.23   1.00   0.45
## review_scores_rating                            -0.37  -0.40   0.45   1.00
## calculated_host_listings_count_entire_homes      0.29   0.37  -0.12  -0.24
## calculated_host_listings_count_private_rooms     0.56   0.58  -0.27  -0.52
```

```
## calculated_host_listings_count_shared_rooms    0.46   0.50  -0.16  -0.32
## reviews_per_month                             -0.22  -0.27   0.41   0.63
##                                              clcltd_hst_lstngs_cnt_n_
## host_response_rate                                                0.09
## host_acceptance_rate                                             -0.31
## host_listings_count                                               0.35
## bathroom_nb                                                      -0.13
## beds                                                             -0.08
## price                                                            -0.07
## minimum_nights                                                    0.32
## maximum_nights                                                    0.30
## availability_30                                                   0.28
## availability_60                                                   0.28
## availability_90                                                   0.29
## availability_365                                                  0.37
## number_of_reviews                                                -0.12
## review_scores_rating                                             -0.24
## calculated_host_listings_count_entire_homes                       1.00
## calculated_host_listings_count_private_rooms                      0.11
## calculated_host_listings_count_shared_rooms                       0.16
## reviews_per_month                                                -0.22
##                                              clcltd_hst_lstngs_cnt_p_
## host_response_rate                                                0.14
## host_acceptance_rate                                             -0.62
## host_listings_count                                               0.96
## bathroom_nb                                                      -0.06
## beds                                                             -0.29
## price                                                            -0.57
## minimum_nights                                                    0.46
## maximum_nights                                                    0.47
## availability_30                                                   0.56
## availability_60                                                   0.55
## availability_90                                                   0.56
## availability_365                                                  0.58
## number_of_reviews                                                -0.27
## review_scores_rating                                             -0.52
## calculated_host_listings_count_entire_homes                       0.11
## calculated_host_listings_count_private_rooms                      1.00
## calculated_host_listings_count_shared_rooms                       0.60
## reviews_per_month                                                -0.36
##                                              clcltd_hst_lstngs_cnt_s_ rvws__
## host_response_rate                                                0.16   0.01
## host_acceptance_rate                                             -0.19   0.36
## host_listings_count                                               0.63  -0.39
## bathroom_nb                                                       0.05   0.07
## beds                                                             -0.12   0.10
## price                                                            -0.41   0.19
## minimum_nights                                                    0.27  -0.40
## maximum_nights                                                    0.47  -0.29
## availability_30                                                   0.47  -0.28
## availability_60                                                   0.46  -0.26
## availability_90                                                   0.46  -0.22
## availability_365                                                  0.50  -0.27
## number_of_reviews                                                -0.16   0.41
## review_scores_rating                                             -0.32   0.63
## calculated_host_listings_count_entire_homes                       0.16  -0.22
```

```
## calculated_host_listings_count_private_rooms                          0.60  -0.36
## calculated_host_listings_count_shared_rooms                           1.00  -0.25
## reviews_per_month                                                    -0.25   1.00
```

First we see that host_listings_count is correlated with availability, calculated_host_listings and host_acceptance_rate so we remove those variables. It also has decent correlation with maximum_nights and review_scores_rating. Due to the higher amount of varialbes we have we decide to also remove those two variablles. The other variables have less correlation and more predictive power.

Next bathroom_nb is highly correlated with beds we will remove bathroom_nb as beds is more indicative of the type of property.

```
numeric <- dplyr::select(df2, c('host_response_rate',
                                'host_listings_count',
                                'beds', 'price', 'minimum_nights',
                                'number_of_reviews',
                                "reviews_per_month"))

corr <- cor(numeric)
colnames(corr) <- abbreviate(colnames(corr), 6)

M <- corr
print(round(M, 2))
```

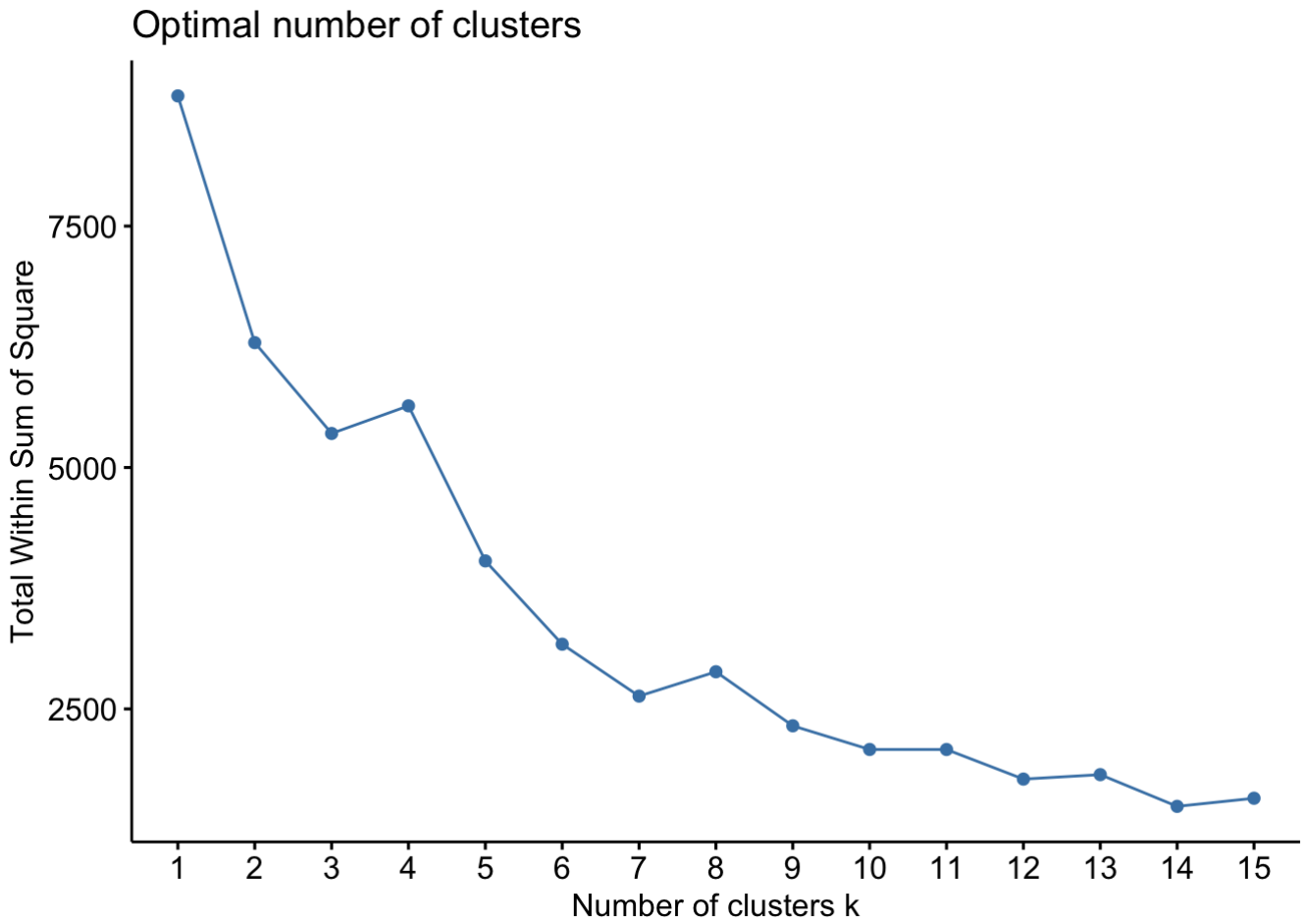
```
##                     hst_r_ hst_l_  beds price mnmm_n nmbr__ rvws__
## host_response_rate    1.00   0.14 -0.03 -0.21   0.03   0.03   0.01
## host_listings_count   0.14   1.00 -0.29 -0.55   0.49  -0.29  -0.39
## beds                 -0.03  -0.29  1.00  0.45  -0.27   0.17   0.10
## price                -0.21  -0.55  0.45  1.00  -0.35   0.20   0.19
## minimum_nights        0.03   0.49 -0.27 -0.35   1.00  -0.24  -0.40
## number_of_reviews     0.03  -0.29  0.17  0.20  -0.24   1.00   0.41
## reviews_per_month     0.01  -0.39  0.10  0.19  -0.40   0.41   1.00
```

All variables have low correlation now so we go to the next step normalize the data and finding the optimal number of clusters. We choose the z-score to scale the data.

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve
3WBa
```

```
numeric <- lapply(numeric,scale) %>% data.frame()

fviz_nbclust(numeric, kmeans, k.max = 15, method ="wss")
```



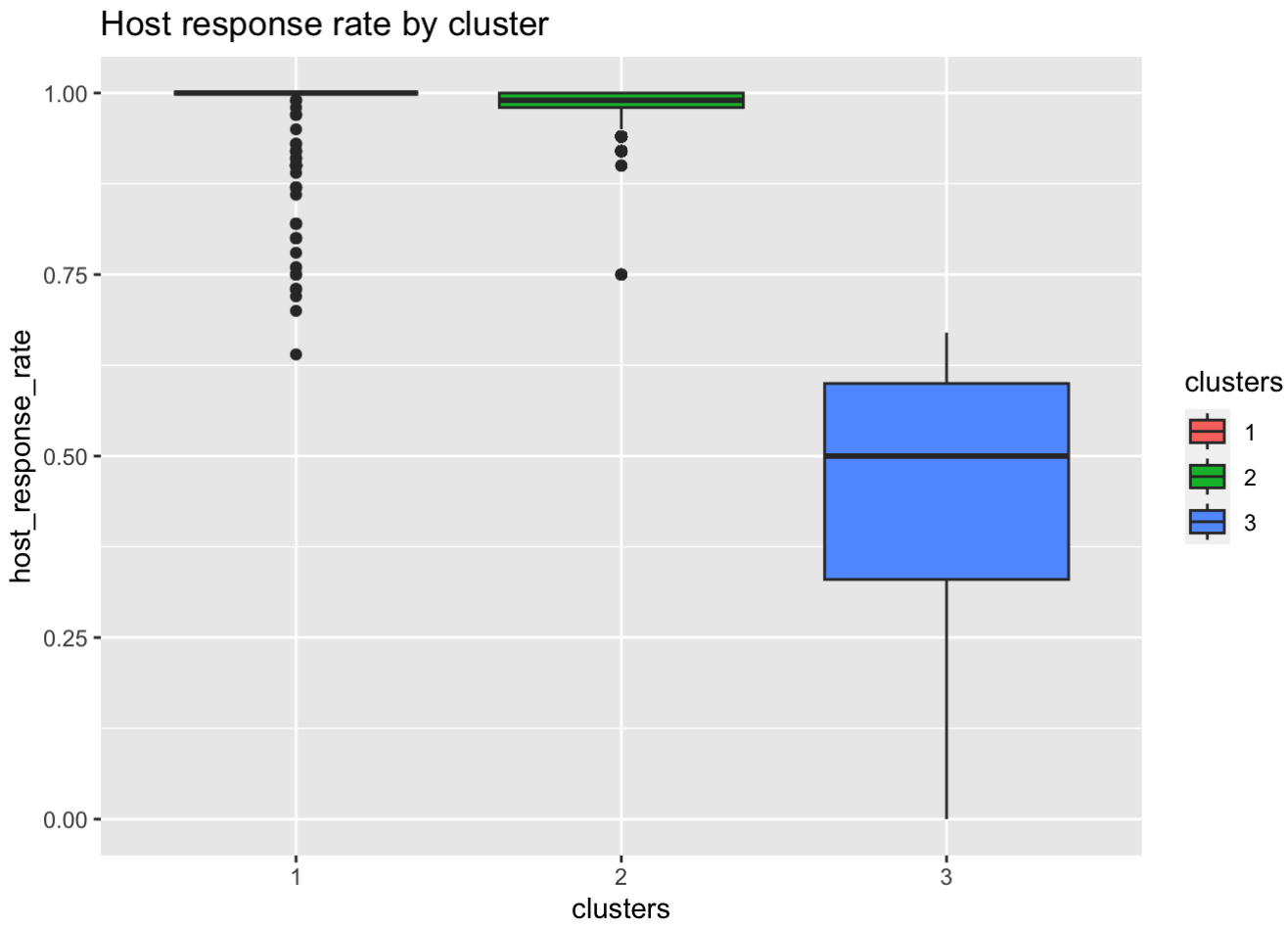
The error stopped going down temporarily at 5 clusters, it could mean that the model is starting to fit noise after 5 clusters. We will stick with 4 clusters, it will also make the interpretation easier. We build the model and assign to its respective cluster each property

```
set.seed(235)
model <- kmeans(numeric, centers = 3, nstart = 100)
df2$clusters <- model$cluster
df2$clusters <- as.factor(df2$clusters)
```

III.

Now let us create visualizations for the clusters to understand what they represent.

```
library(ggplot2)

ggplot(df2) +
  geom_boxplot(aes(x=clusters, y = host_response_rate, fill = clusters)) +
  labs(title="Host response rate by cluster") +
  scale_x_discrete()
```



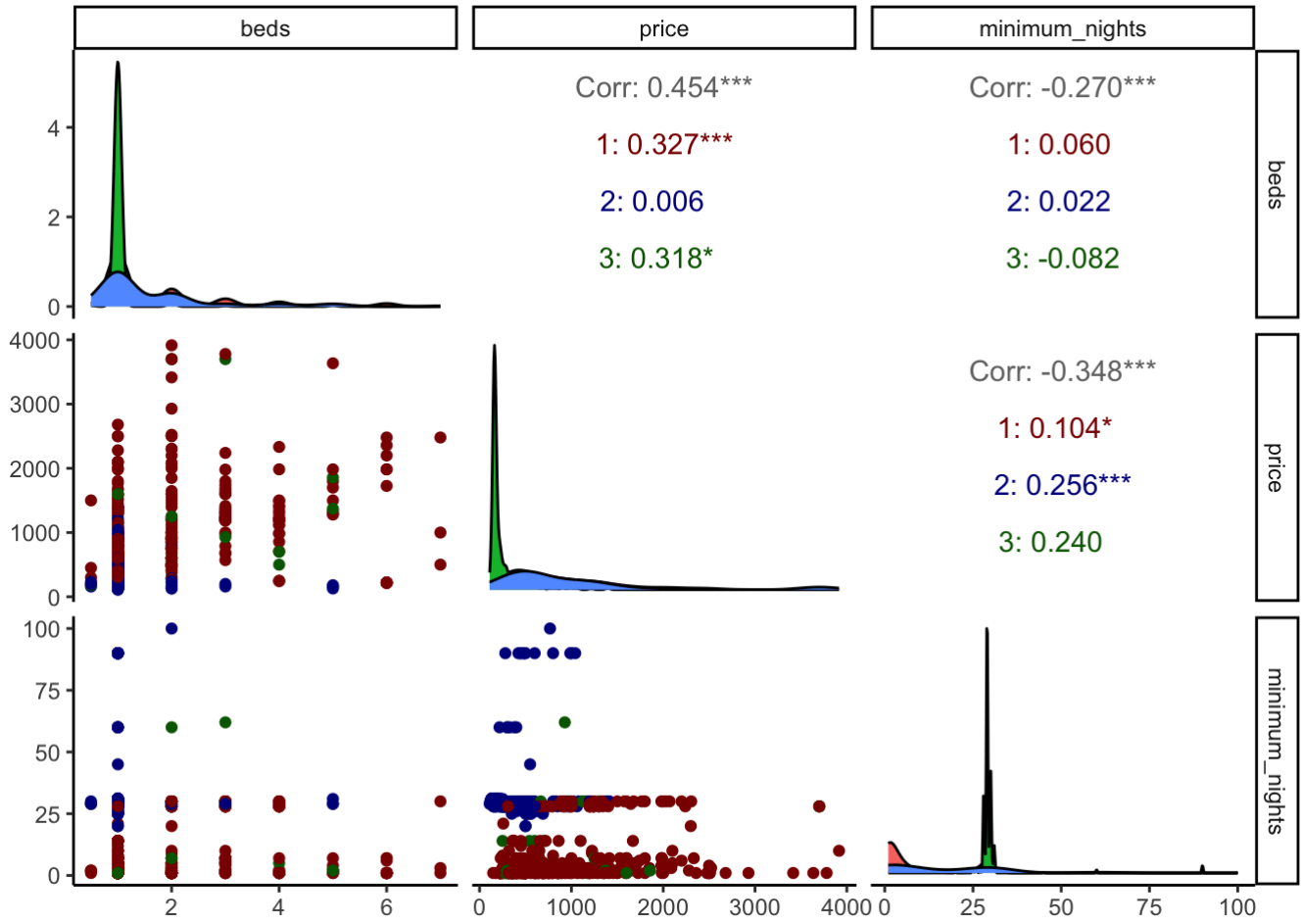
Cluster 3 has a very low response rate compared to the others, all other clusters have most of their response rate over 0.95. Cluster 1 and 2 have almost all their response rate around 1 except for outliers.

```
numeric$clusters <- df2$clusters
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
ggpairs(df2, columns = c('beds', 'price', 'minimum_nights'),
        aes(color = clusters)) +
  theme_classic() +
 scale_color_manual(values = c("darkred", "darkblue", "darkgreen"),
                    labels = c("Cluster 1", "Cluster 2", "Cluster 3"))
```

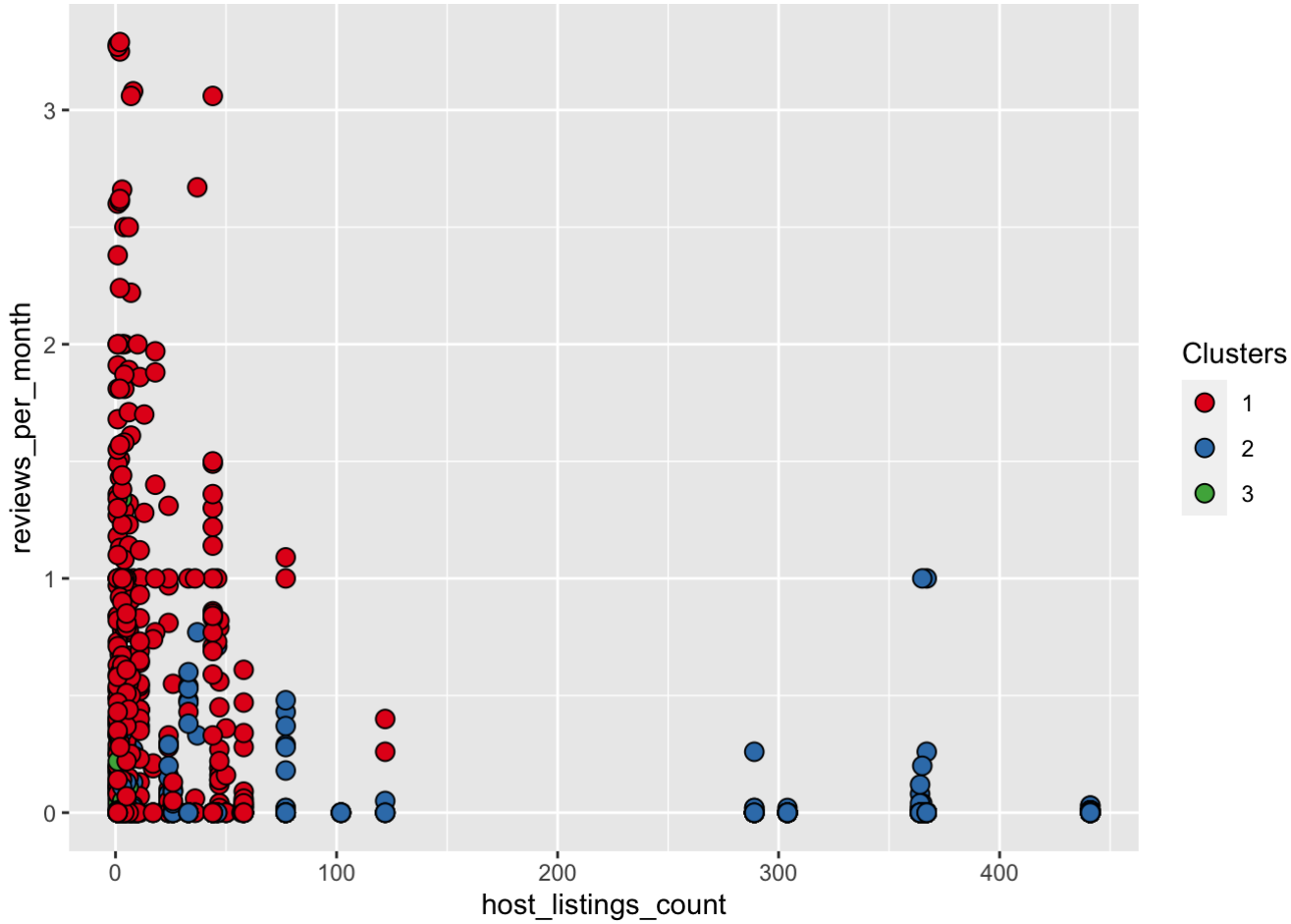


Here we can see Cluster 1 has high price for most of its properties with some on the lower end and also low number of minimum nights meaning it is more oriented towards short stays with high prices.

Cluster 2 has medium to high minimum nights with low prices and low amounts of beds. It is geared towards cheap long stays for only one or a few people.

Cluster 3 has average price, low amount of minimum nights and average amount of beds.

```
ggplot(df2, aes(x = host_listings_count, y = reviews_per_month, fill = clusters)) +
 scale_fill_brewer(palette = "Set1", name = "Clusters") +
 geom_point(shape = 21, color = "black", size = 3)
```

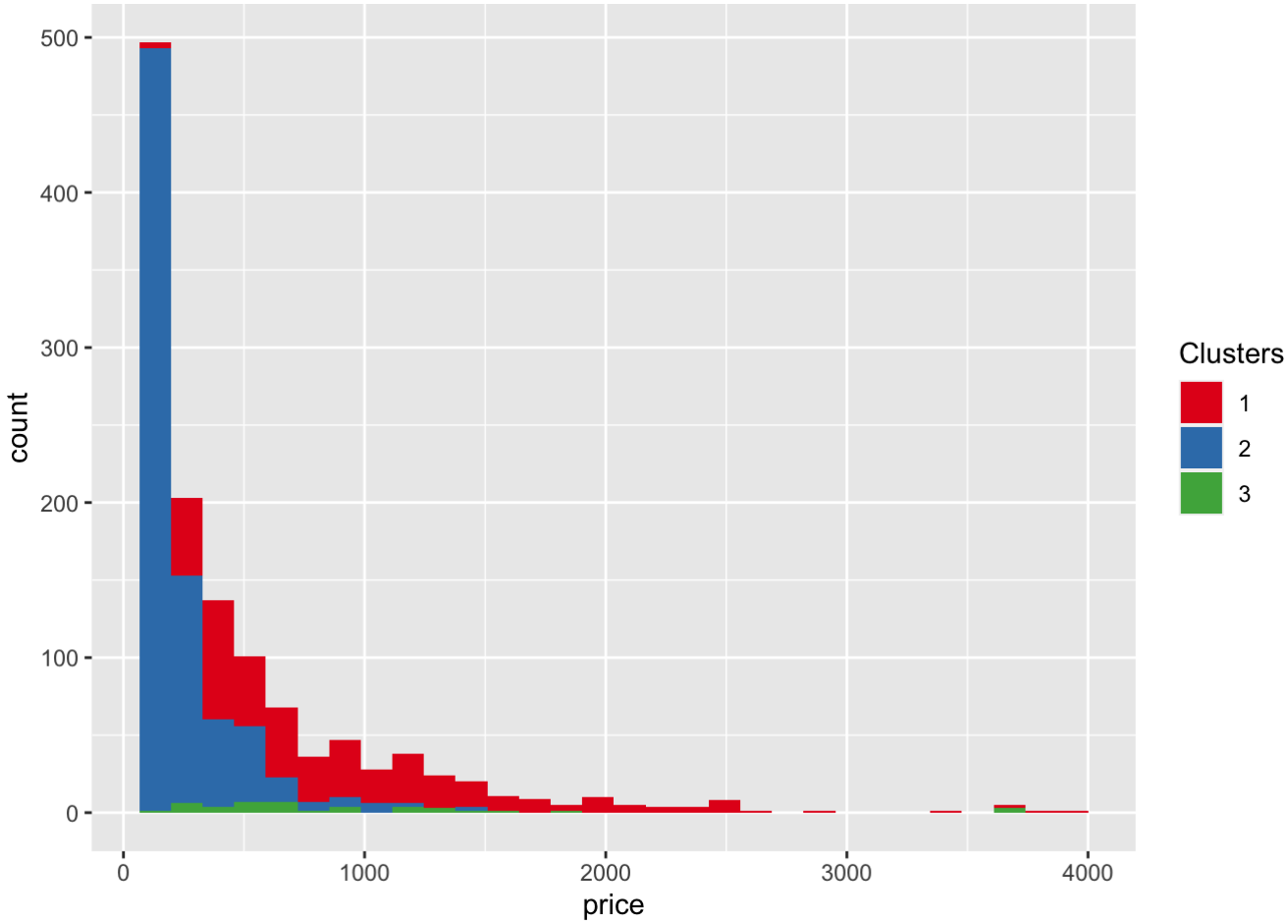


From this scatterplot we can clearly see that Cluster 2 has usually the highest amount of host listings count. It has half of its apartments that has average host listings count around 60 while the other half has very high amount of listings counts around 350. Cluster 1 and 3 have a lower amount of host_listings_count with a high density under 10.

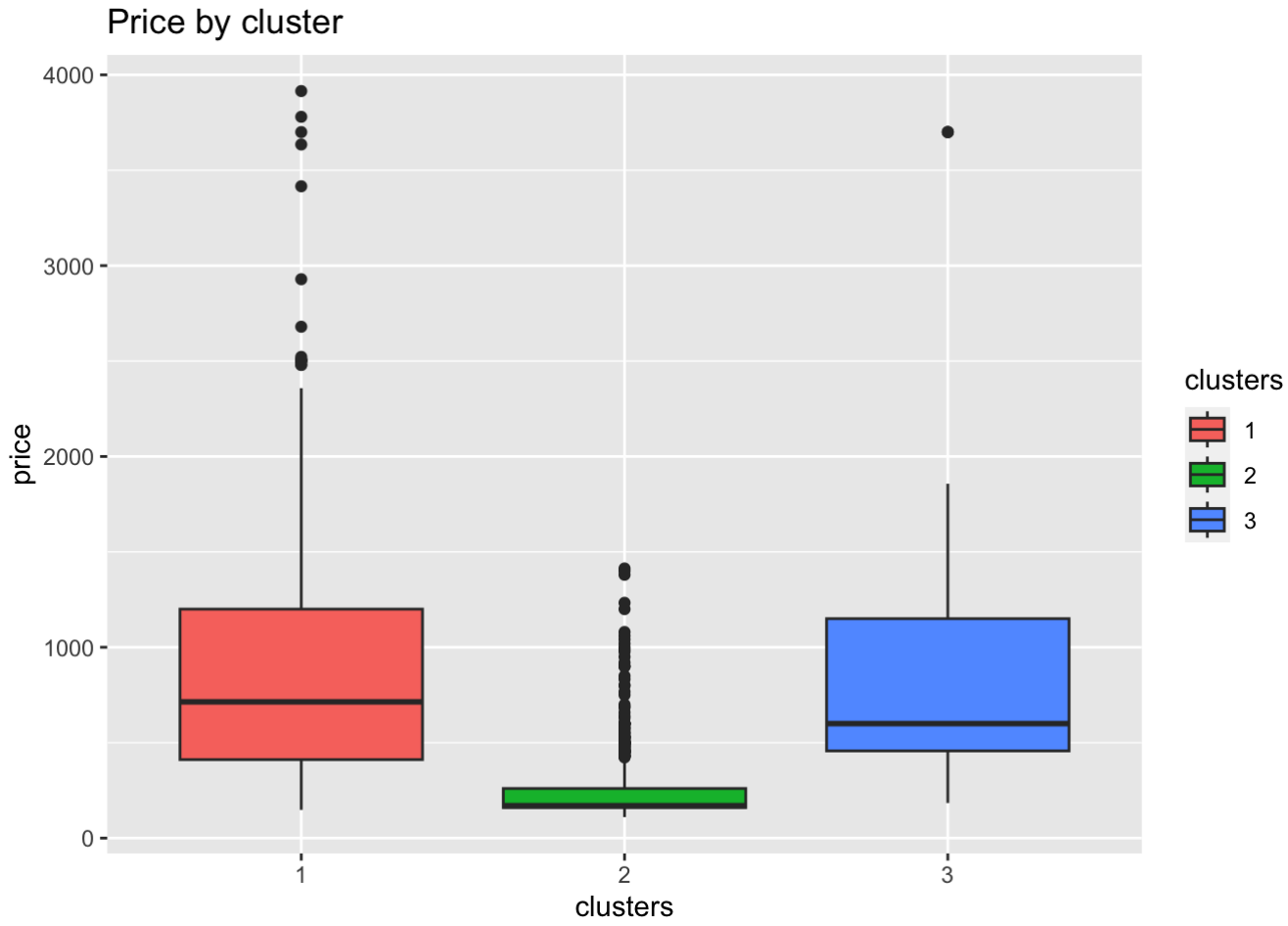
Cluster 1 has higher number of reviews compared to the other two.

```
ggplot(df2, aes(x = price, fill = clusters)) +
 geom_histogram() +
 scale_fill_brewer(palette = "Set1", name = "Clusters")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(df2) +
  geom_boxplot(aes(x=clusters, y = price, fill = clusters)) +
  labs(title="Price by cluster") +
  scale_x_discrete()
```



In this boxplot we can clearly see that the three clusters are different in prices. CLuster 2 has the lowest prices around 300 but with many outliers that are around 1000. Cluster 1 has most of its prices between 500 and 1200 with some outliers going the highest around $ 3700.

Finally Cluster 3 has its price distribution close to cluster 1 but with only one outlier and its median lower.

II.

In this k-means clustering model first filtered through numerical variables then we calculated the correlation between those variables and removed the variables that high correlation according to how meaningful it is for information.

Then we removed outliers using the 99% percentile as we've seen from boxplots previously that most outliers were very far from the majority of the data.

Then we scaled the data to prepare for clustering.

Finally we chose the number of clusters using the elbow method stopping at an anomaly when the error did not go down when the number of clusters went up.

After building the model we built visualizations to get insights about what each cluster represents. Here are how we would define and name each cluster according to this :

Cluster 1: Popular Short Stays

They correspond to high price range with some lower-priced outliers. Low number of minimum nights, indicating a focus on short stays. Higher number of reviews compared to the other clusters meaning that they are well known. Lower host listings count, mostly under 10 so owners are not corporate.

Cluster 2: Budget Long-Stay

With low prices, high minimum nights, and low bed counts, this cluster seems geared towards cheaper long-term stays for 1-2 people. Listings are also split between average and high host counts.

Cluster 3: Mid-Market Mixed-Stay The moderate price range, average minimum nights and bed counts suggest this cluster is a mix of both short and long stays at medium price points. Lower host counts than Cluster 2.

Conclusion

First we removed unnecessary columns, identified missing values and used various techniques such as prediction and imputation to clean the data. This part was particularly difficult since it implied many judgement calls on the each variable and understanding prior to building the models what we would need to make it as easy as possible. Then we successfully identified variables with many outliers using boxplots for the next phases, as those can heavily reduce efficiency of the models.

Next we tried visualizations to get a first understanding of the data. This gave us many insights on different types of property owners such as individuals and management companies. We understood patterns in room types for different strategies for budget travelers or the owners trying large portfolios for investment, higher investment owners going for more average pricings. Finally the boxplots and histograms showed how most pricing are lower than which is not a surprise as most airbnb are rooms for big cities like Hong Kong. Finally the Geopsatial mapping of our neighborhood showed the distribution of entire homes on the outskirts while shared rooms were in the district. It also showed different landmarks such as commercial centers, transport and cultural sites.

The first model we built was the multilinear model where we had to make hard choices in choosing the variables. Classic feature selection techniques allowed us find the right variables. The most interesting aspect of this part was how through applying the log function to the price output variable our visualizations showed that linear relationships would appear with predictors. This hinted to the model being more accurate with the log of the price which proved to be true. The final model had a decent accuracy with a margin of error. We had many variables and finding the right pricing is a very difficult task even for marketing experts so we still considered it a success.

Next we built multiple classification models. Again required good intuition to choose the variables. First we tried predicting the availability of a washer by looking at the most similar properties. We believed that this amenity is a strong predictor of the type of owner as it is more common for washers to be in public spaces for renters. We got good prediction results on the testing set indicating a rigorous model. The other model used was naive Bayes to predict average review rating. This variable was problematic because most properties did not have reviews. But we used some amenities and other relevant variables to keep for prediction and we obtained model with a decent level of prediction of the outcome.

Two last classification models were built. First a classification tree where we optimized its complexity to get the best result on the validation set. The model was good at predicting when a property will not be instantly bookable but less so at predicting when it is instantly bookable. Second a clustering model where we identified three distinct clusters of properties one for expensive short stays another for budget long-stays and a last one for mix market, mix stays. This clustering gives another way of understanding the data.

In conclusion, the greatest challenge of this massive dataset was to find the right variables for the models which required good understand and intuition of what the data shows and what we are trying to achieve. The visualizations and the various techniques to reduce the amount of features were very useful in finding the right direction to achieve this task. In the end all our models gave good predictions or important insights on the data and could be useful for any company looking to exploit the rental market in Hong Kong.