Final Team Project - AirBnb Boston Data

Code

Step I: Data Preparation & Exploration

Read data into your local environment

Hide

```
df <- read.csv("metad699_train.csv")
View(df)
library(dplyr)
```

```


Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

Hide

```
library(tidyr)
library(tidyverse)
```

```
[37m── [1mAttaching packages [22m ─────────────────────────────────────────── tidyvers
e 1.3.0 ── [39m
[37m [32m✓ [37m [34mggplot2 [37m 3.2.1     [32m✓ [37m [34mpurrr  [37m 0.3.3
[32m✓ [37m [34mtibble [37m 2.1.3     [32m✓ [37m [34mstringr [37m 1.4.0
[32m✓ [37m [34mreadr  [37m 1.3.1     [32m✓ [37m [34mforcats [37m 0.4.0 [39m
[37m── [1mConflicts [22m ──────────────────────────────────────────── tidyverse_conf
licts() ──
[31mx [37m [34mdplyr [37m:: [32mfilter() [37m masks [34mstats [37m::filter()
[31mx [37m [34mdplyr [37m:: [32mlag() [37m    masks [34mstats [37m::lag() [39m
```

Hide

```
library(ggplot2)
library(caret)
```

```
Loading required package: lattice

Attaching package: ‘caret’

The following object is masked from ‘package:purrr’:

    lift
```

Hide

```
library(plyr)
```

```
------------------------------------------------------------------------------------------------------------
You have loaded plyr after dplyr - this is likely to cause problems.
If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
library(plyr); library(dplyr)
------------------------------------------------------------------------------------------------------------

Attaching package: ‘plyr’

The following object is masked from ‘package:purrr’:

    compact

The following objects are masked from ‘package:dplyr’:

    arrange, count, desc, failwith, id, mutate, rename, summarise, summarize
```

Hide

```
library(forecast)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
This is forecast 8.11
  Want to meet other forecasters? Join the International Institute of Forecasters:
  http://forecasters.org/
```

Hide

```
boston <- filter(df, city=="Boston")
```

I. Missing Values

Code

```
[1] "We first used the anyNA function to determine if we had any missing data. Upon filtering the data with our s
elected city of Boston, we were able to observe a significant number of blank cells and missing values. We decide
d to find and use the median values as replacements and replaced blank spaces with NA's. Overall, we ended up pre
serving our full filtered data with 3468 observations and 29 variables. Upon preserving all our data for numerica
l categories, we deleted unnecessary columns and further cleaned our data frame where no missing NA's were presen
t in colSums. We re-named our finalized copy as 'boston1'."
```

Hide

```
anyNA(boston)
```

```
[1] TRUE
```

Hide

```
# Explore missing values
View(boston)
colSums(is.na(boston))
```

```
                   id             log_price         property_type             room_type 
                    0                     0                     0                     0 
            amenities          accommodates             bathrooms              bed_type 
                    0                     0                     6                     0 
  cancellation_policy          cleaning_fee                  city           description 
                    0                     0                     0                     0 
         first_review  host_has_profile_pic host_identity_verified    host_response_rate 
                    0                     0                     0                     0 
           host_since      instant_bookable           last_review              latitude 
                    0                     0                     0                     0 
            longitude                  name          neighbourhood     number_of_reviews 
                    0                     0                     0                     0 
 review_scores_rating         thumbnail_url               zipcode              bedrooms 
                  648                     0                     0                     3 
                 beds 
                    2 
```

Hide

```
# Explore median values for missing column without factoring NA's
median(boston$review_scores_rating, na.rm = TRUE)
```

```
[1] 96
```

Hide

```
median(boston$bathrooms, na.rm = TRUE)
```

```
[1] 1
```

Hide

```
median(boston$bedrooms, na.rm = TRUE)
```

```
[1] 1
```

Hide

```
median(boston$bathrooms, na.rm = TRUE)
```

```
[1] 1
```

Hide

```
median(boston$bedrooms, na.rm = TRUE)
```

```
[1] 1
```

Hide

```
median(boston$beds, na.rm=TRUE)
```

```
[1] 1
```

Hide

```
# replace all NA's
boston[boston== ""] <-NA
# replace all NA's with median value
boston$review_scores_rating[is.na(boston$review_scores_rating)] <- median(boston$review_scores_rating, na.rm = TR
UE)
boston$host_response_rate <- as.numeric(sub("%","",boston$host_response_rate))/100
boston$host_response_rate[is.na(boston$host_response_rate)] <- median(boston$host_response_rate, na.rm=TRUE)
boston$beds[is.na(boston$beds)] <- median(boston$beds, na.rm = T)
boston$bathrooms[is.na(boston$bathrooms)] <- median(boston$bathrooms, na.rm=T)
boston$bedrooms[is.na(boston$bedrooms)] <- median(boston$bedrooms, na.rm=T)
colSums(is.na(boston))
```

```
                   id            log_price        property_type            room_type
                    0                    0                    0                    0
            amenities         accommodates            bathrooms             bed_type
                    0                    0                    0                    0
  cancellation_policy         cleaning_fee                 city          description
                    0                    0                    0                    0
         first_review host_has_profile_pic host_identity_verified   host_response_rate
                  621                    0                    0                    0
           host_since     instant_bookable          last_review             latitude
                    0                    0                  621                    0
            longitude                 name        neighbourhood    number_of_reviews
                    0                    0                    0                    0
 review_scores_rating        thumbnail_url              zipcode             bedrooms
                    0                  134                   26                    0
                 beds
                    0
```

Hide

```
# delete unecessary information
boston1 <- boston[-c(13, 11, 19, 26, 27)]
View(boston1)
colSums(is.na(boston1))
```

```
                     id              log_price          property_type              room_type 
                      0                      0                      0                      0 
              amenities           accommodates              bathrooms               bed_type 
                      0                      0                      0                      0 
    cancellation_policy           cleaning_fee            description   host_has_profile_pic 
                      0                      0                      0                      0 
 host_identity_verified     host_response_rate             host_since       instant_bookable 
                      0                      0                      0                      0 
               latitude              longitude                   name          neighbourhood 
                      0                      0                      0                      0 
      number_of_reviews   review_scores_rating               bedrooms                   beds 
                      0                      0                      0                      0 
```

II. Summary Statistics

Code

```
[1] "In our selected data frame, we wanted to observe the true nightly price from log_price to understand the tru
e dollar format. Upon running the summary of boston1 we noticed the following observations for the selected varia
bles:\n\n\t\tReview score ratings: Out of 100 being the highest score for review ratings from \t\t\tcustomers, 20
was the lowest. The median was 96 and the mean was 94.05 which makes the distribution of review ratings negativel
y skewed. When we further observe the standard deviation of 7.327312, we can conclude that our review ratings are
very close to the mean.\n\n\t\tLog price/Nightly price: We know that log price and nightly price are practically
the same. The max rental price for an Airbnb in Boston is $1,400 a night while the lowest is $17. The median pric
e is $136 while the mean is $165.50 which means that \tmedian is low and mean is high. In contrast to log price f
orm, the median is higher while the mean is lower. This means that the dollar form of our nightly price data is
\tpositively skewed while in log form, the price is negatively skewed. A possibility \tfor why this happens is th
e normalization of data where the price in normal format can be far spread out. This makes sense if we were to ob
serve the standard deviation for both prices. In log format, the standard deviation is close to 0 which \t\tmeans
that all the data point are close to its mean while the nightly price has a standard deviation farther apart from
mean.  \n"
```

Hide

```
boston1$nightly_price <- exp(boston1$log_price) # nightly price converion from log
# Selected summary of statistics
summary(boston1$review_scores_rating)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  20.00   92.00   96.00   94.05   98.00  100.00 
```

Hide

```
summary(boston1$log_price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.833   4.382   4.913   4.884   5.298   7.244 
```

Hide

```
summary(boston1$nightly_price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   17.0    80.0   136.0   165.6   200.0  1400.0 
```

Hide

```
sd(boston1$review_scores_rating)
```

```
[1] 7.327312
```

Hide

```
sd(boston1$log_price)
```

```
[1] 0.6646924
```

Hide

```
sd(boston1$nightly_price)
```

```
[1] 128.8892
```

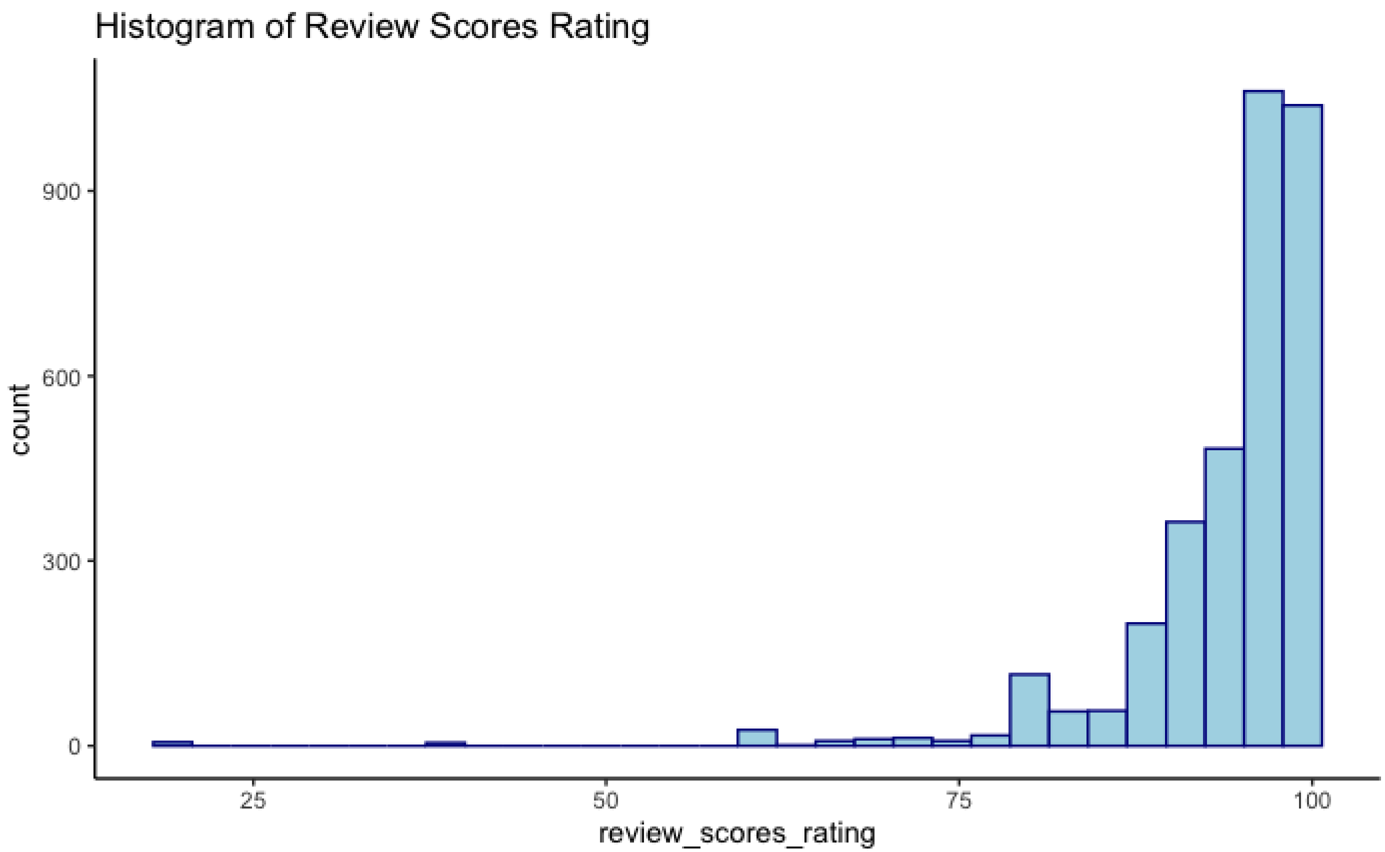
III. Visualization

Code

```
[1] "For histogram and scatter plot we saw that our scores rating remained the same. Using the Histogram, we can
see that the majority of our data is skewed to the right where the scores tend to be above 75%. Only few outliers
exist where the review scores are less than 50%. The same can be confirmed with the Scatter Plot where the bulk
of the reviews are rated at above 80%. This indicates that the reviews for AirBnb rentals in Boston are positive
\n\nWe further expanded the exploration of our data by choosing neighborhood and log price as variables. If there
is one thing that influences someone to book a room, house, or apartment, it is the price you pay to book your re
ntal. In our bar plot and violin plot, we can observe the log price per neighborhood. In the given bar plot, we s
ee that the Allston-Brighton carries the bulk of the rentals based on prices. This means that those who rent thro
ugh AirBnb, would rent the most in that area with the given price range. The Violin Plot gives a better indicatio
n of where that range lies. Based on the distribution of prices for Allston-Brighton, the bulk for log price lies
slightly above 3 and up to about 5. We can conclude that the area is a frequent Airbnb hotspot due to its lower p
rices.\n\nWhen we examine our boxplot, in reference to review scores ratings and cancellation policies, the flexi
ble a rental is, the higher the review will be. We can see that as the box plot becomes larger as the cancellatio
n policy becomes more strict.\n"
```

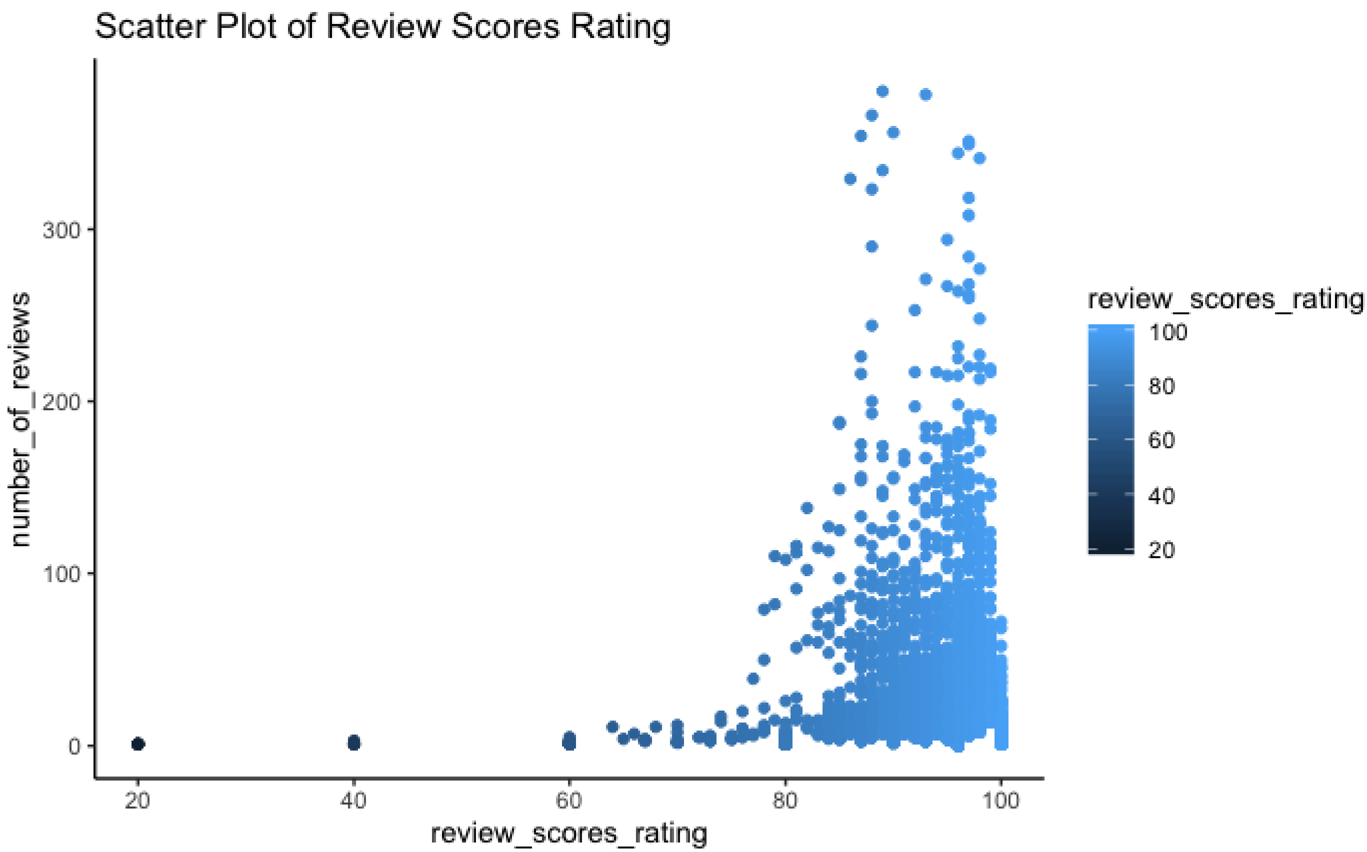
Hide

```
ggplot(boston1, aes(x=review_scores_rating)) +
  geom_histogram(color="darkblue", fill="lightblue") +
  labs(title="Histogram of Review Scores Rating") +
  theme_classic()
```



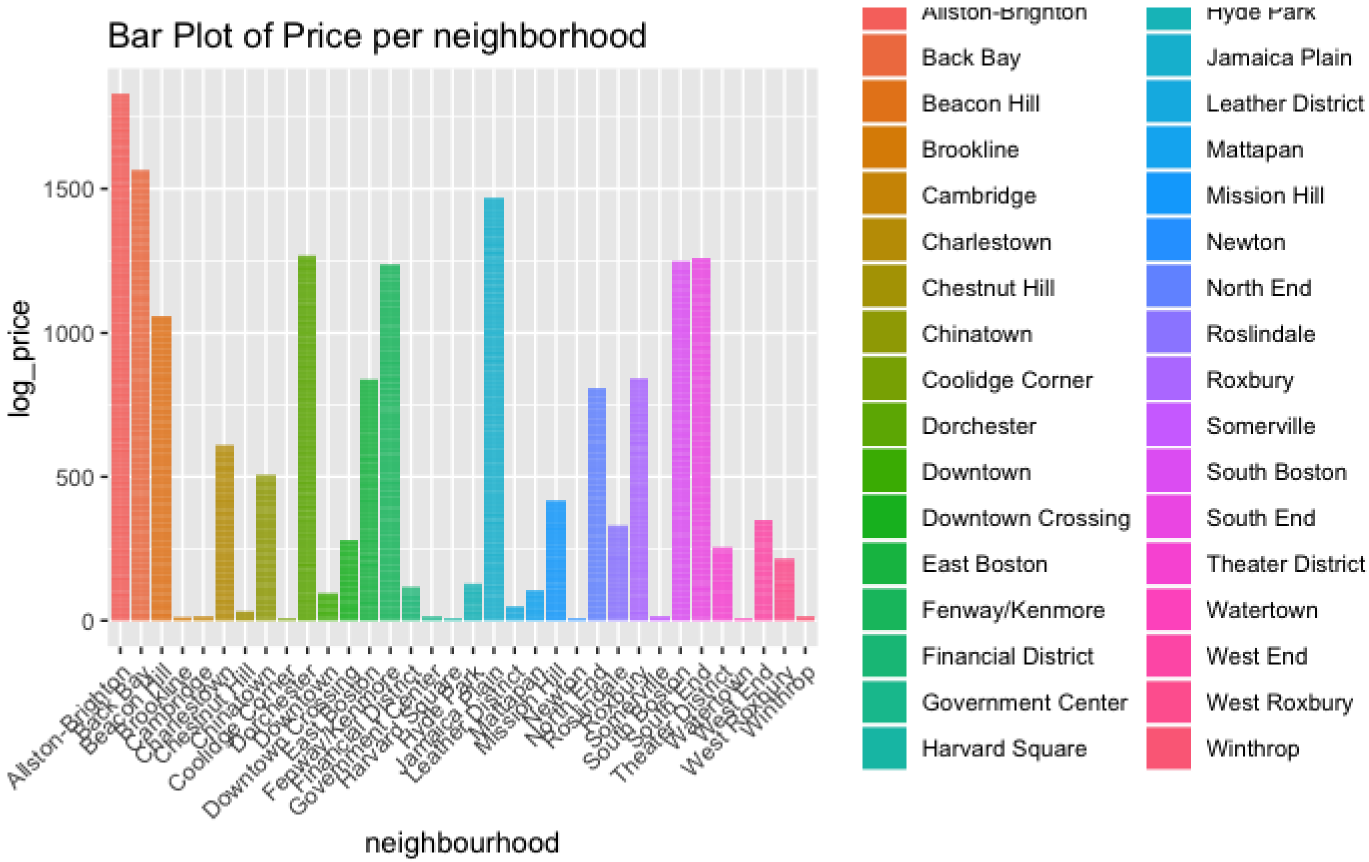
Hide

```
ggplot(boston1, aes(x=review_scores_rating, y=number_of_reviews, color=review_scores_rating)) +
  geom_point() +
  labs(title="Scatter Plot of Review Scores Rating") +
  theme_classic()
```



Hide

```
ggplot(boston1, aes(x=neighbourhood, y=log_price, fill=neighbourhood)) +
  geom_bar(stat = "identity") +
  labs(title="Bar Plot of Price per neighborhood") +
  theme(axis.text.x = element_text(angle=45, hjust=1))
```



Hide

```
install.packages("Hmisc")
```

```
Installing package into '/Users/josemartinez/Library/R/3.6/library'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/Hmisc_4.4-0.tgz'
Content type 'application/x-gzip' length 3146788 bytes (3.0 MB)
==================================================
downloaded 3.0 MB
```

```
The downloaded binary packages are in
    /var/folders/6v/wsr694r57n9dfsdxftfdysbh0000gn/T//Rtmp8P9CXE/downloaded_packages
```

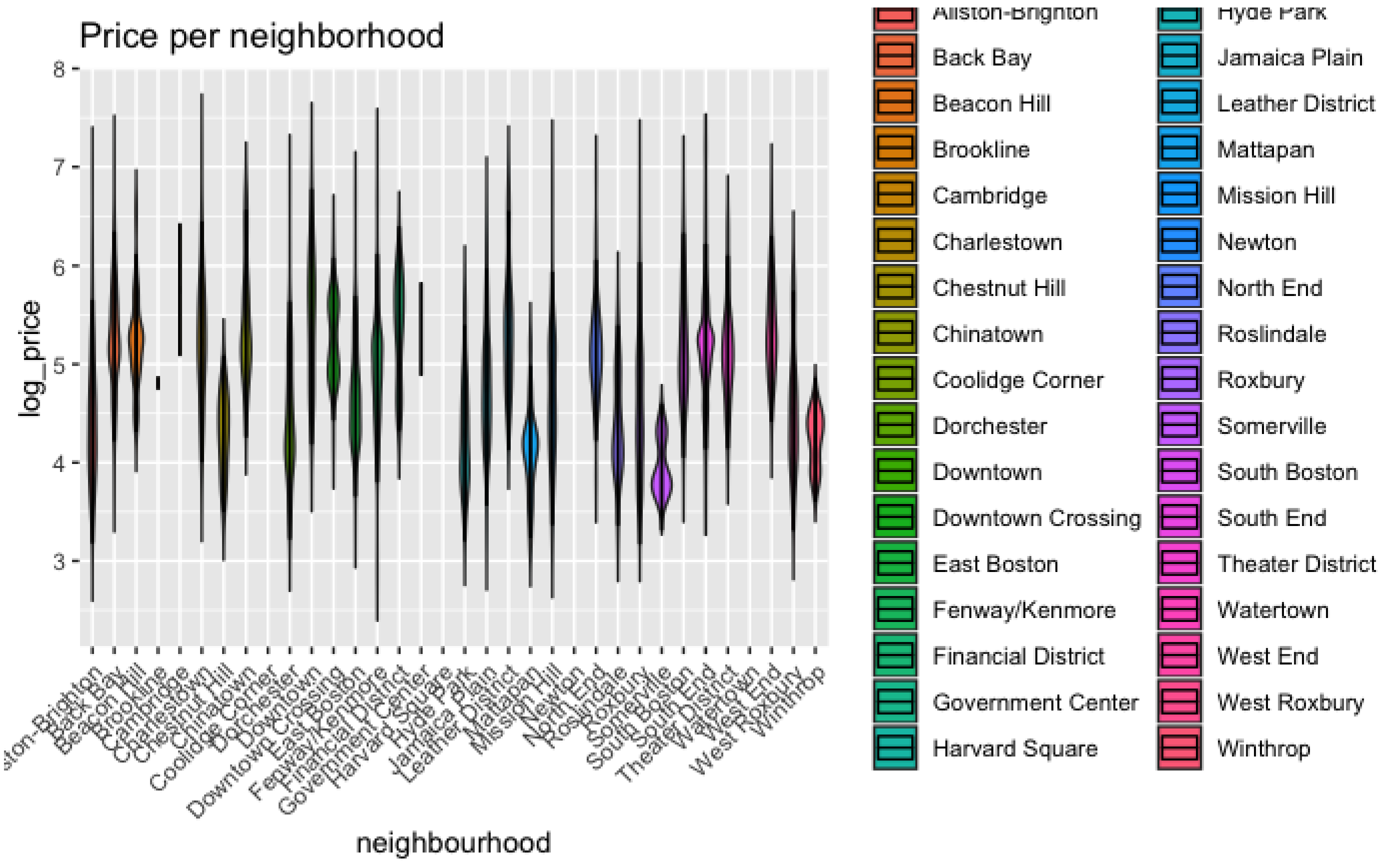
Hide

```
Violin <- ggplot(boston1, aes(neighbourhood, y=log_price, fill=neighbourhood)) +
  geom_violin(trim=FALSE) +
  stat_summary(fun.data="mean_sdl", mult=1, geom="crossbar", width=0.04 ) +
  labs(title = "Price per neighborhood") + theme(axis.text.x = element_text(angle=45, hjust=1))
```

```
Ignoring unknown parameters: mult
```

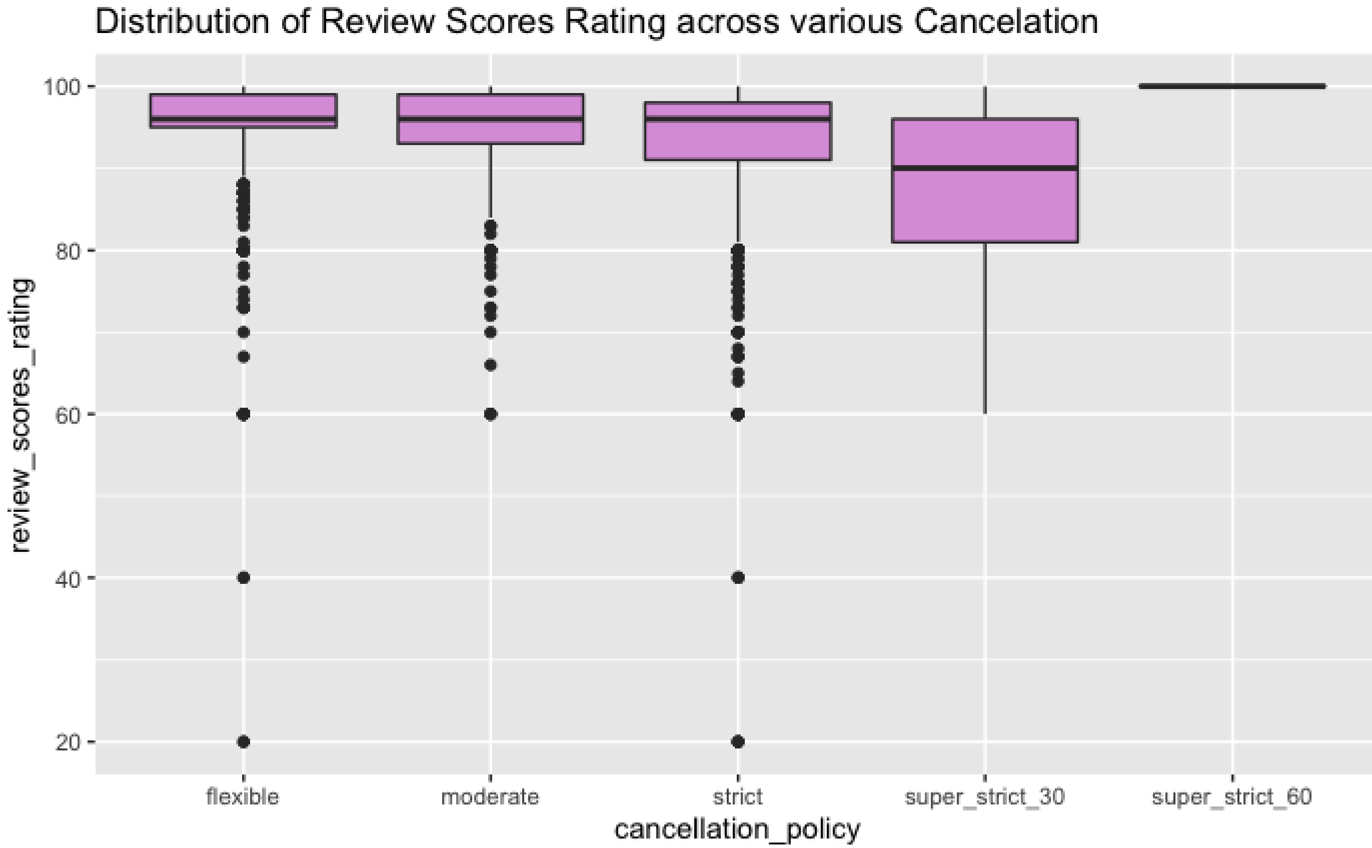
Hide

```
Violin
```



Hide

```
ggplot(boston1, aes(cancellation_policy, review_scores_rating)) + geom_boxplot(fill="plum") +
  labs(title="Distribution of Review Scores Rating across various Cancelation ")
```



Step II: Prediction

Code

```
[1] "We began by setting up a correlation table and looking at the variables that were  heavily correlated with e
ach other. Due to the numerous amounts of variables that were provided in the boston1 dataframe, we knew that our
best option was to select a multiple linear regression model to determine our prediction. We used the sapply func
tion to vector all columns and create a matrix.  Based on the correlation table that was created without excludin
g any data, we saw that log_price and nightly_price were completely correlated with each other. We also noticed t
hat beds, bedrooms, and accommodates were heavily correlated. Overall, we decided to remove id, nightly_price, be
ds, and bedrooms to prevent multicollinearity. The below heatmap, indicates that there is no issue of multicollin
earity.\n\nWe further expanded the selection of our variables by using the backward elimination method in our mul
tiple linear regression model. We used the 60/40 method to slice our data and train our model before validating i
t. Upon running the backward elimination, we saw that our recommended variables were narrowed down to 14 with an
intercept of 1.20057241. If we were to determine our log_price for any given coefficient such accommodates as dis
played in our regression summary, our regression formula would be as follows,\n\nlog_price = 1.200 + 0.0819 * acc
ommodates\n\nAssume you want to accommodate for 3 people, the equation would be as follows,\n\nlog_price = 1.200
+ 0.0819 * 3\nlog_price = 1.4457\n\nThe r-squared for our model is 0.5994. This means that close to 60% of our se
lected variables points would fit on the regression line. Our RMSE is 0.4444 which measures the difference betwee
n predicted values and actual values. The closer the number is to 0, the better. \n"
```

Hide

```
# normalize all data points
install.packages("reshape")
```

```
Installing package into ‘/Users/josemartinez/Library/R/3.6/library’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/reshape_0.8.8.tgz'
Content type 'application/x-gzip' length 172673 bytes (168 KB)
==================================================
downloaded 168 KB
```

```


The downloaded binary packages are in
    /var/folders/6v/wsr694r57n9dfsdxftfdysbh0000gn/T//RtmplIEjsq/downloaded_packages
```

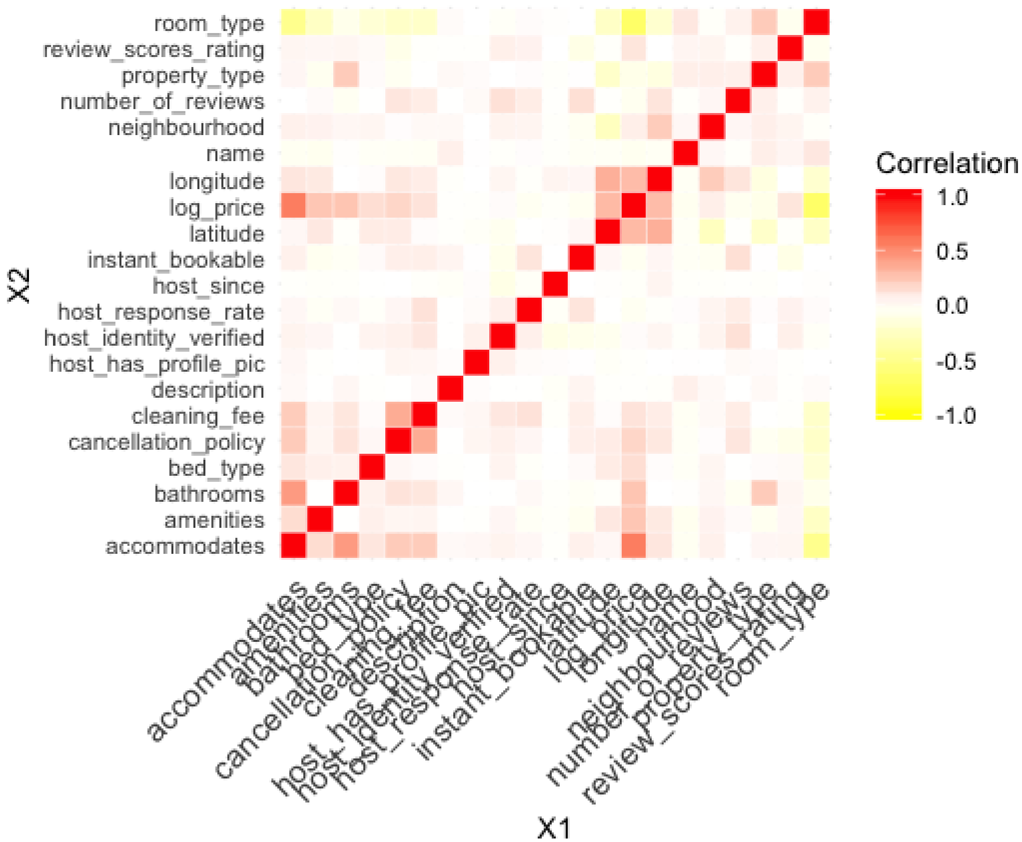
Hide

```
library(reshape)
```

```
Attaching package: 'reshape'

The following objects are masked from 'package:plyr':

    rename, round_any

The following objects are masked from 'package:tidyr':

    expand, smiths

The following object is masked from 'package:dplyr':

    rename
```

Hide

```
bos <- boston1
bos <- bos[-c(1, 23, 24 ,25)]
must_convert <- sapply(bos, is.factor)
m2 <- sapply(bos[, must_convert], unclass)
bos <- cbind(bos[,!must_convert], m2)
table <- cor(bos)
melted_table <- melt(table)
# using a heatmap for slected variables
library(ggplot2)
ggplot(data = melted_table, aes(X1, X2, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "yellow", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Correlation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1)) + coord_fixed()
```



Hide

```
468 * 0.60
```

```
[1] 280.8
```

Hide

```
3468 * 0.40
```

```
[1] 1387.2
```

Hide

```
Training <- slice(bos, 1:2080)
Validation <- slice(bos, 2081:3468)
# using the backward elimination method to further finalize our variables
boston_mlr <- lm(log_price~ ., data = Training)
step_boston_mlr <- step(boston_mlr, direction = "backward")
```

```
Start:  AIC=-3562.42
log_price ~ accommodates + bathrooms + cleaning_fee + host_response_rate + 
    latitude + longitude + number_of_reviews + review_scores_rating + 
    property_type + room_type + amenities + bed_type + cancellation_policy + 
    description + host_has_profile_pic + host_identity_verified + 
    host_since + instant_bookable + name + neighbourhood

                         Df Sum of Sq    RSS     AIC
- description             1     0.000 367.69 -3564.4
- name                    1     0.003 367.69 -3564.4
- cancellation_policy     1     0.062 367.75 -3564.1
- host_since              1     0.072 367.76 -3564.0
- host_has_profile_pic    1     0.155 367.84 -3563.5
- host_identity_verified  1     0.342 368.03 -3562.5
<none>                                367.69 -3562.4
- property_type           1     0.577 368.26 -3561.2
- number_of_reviews       1     0.602 368.29 -3561.0
- host_response_rate      1     0.918 368.60 -3559.2
- bed_type                1     0.939 368.62 -3559.1
- neighbourhood           1     0.998 368.68 -3558.8
- review_scores_rating    1     1.676 369.36 -3555.0
- amenities               1     1.861 369.55 -3553.9
- instant_bookable        1     2.040 369.73 -3552.9
- cleaning_fee            1     2.123 369.81 -3552.4
- bathrooms               1     5.416 373.10 -3534.0
- longitude               1     8.399 376.08 -3517.4
- latitude                1    13.090 380.78 -3491.7
- accommodates            1    39.772 407.46 -3350.8
- room_type               1   119.723 487.41 -2978.1

Step:  AIC=-3564.42
log_price ~ accommodates + bathrooms + cleaning_fee + host_response_rate + 
    latitude + longitude + number_of_reviews + review_scores_rating + 
    property_type + room_type + amenities + bed_type + cancellation_policy + 
    host_has_profile_pic + host_identity_verified + host_since + 
    instant_bookable + name + neighbourhood

                         Df Sum of Sq    RSS     AIC
- name                    1     0.003 367.69 -3566.4
- cancellation_policy     1     0.062 367.75 -3566.1
```

```
- host_since               1     0.072 367.76 -3566.0
- host_has_profile_pic     1     0.155 367.84 -3565.5
- host_identity_verified   1     0.342 368.03 -3564.5
<none>                                 367.69 -3564.4
- property_type            1     0.577 368.26 -3563.2
- number_of_reviews        1     0.602 368.29 -3563.0
- host_response_rate       1     0.919 368.60 -3561.2
- bed_type                 1     0.939 368.62 -3561.1
- neighbourhood            1     1.001 368.69 -3560.8
- review_scores_rating     1     1.676 369.36 -3557.0
- amenities                1     1.864 369.55 -3555.9
- instant_bookable         1     2.045 369.73 -3554.9
- cleaning_fee             1     2.127 369.81 -3554.4
- bathrooms                1     5.417 373.10 -3536.0
- longitude                1     8.407 376.09 -3519.4
- latitude                 1    13.097 380.78 -3493.6
- accommodates             1    39.810 407.50 -3352.6
- room_type                1   119.810 487.50 -2979.8

Step:  AIC=-3566.4
log_price ~ accommodates + bathrooms + cleaning_fee + host_response_rate + 
    latitude + longitude + number_of_reviews + review_scores_rating + 
    property_type + room_type + amenities + bed_type + cancellation_policy + 
    host_has_profile_pic + host_identity_verified + host_since + 
    instant_bookable + neighbourhood

                         Df Sum of Sq    RSS     AIC
- cancellation_policy     1     0.062 367.75 -3568.1
- host_since              1     0.072 367.76 -3568.0
- host_has_profile_pic    1     0.155 367.84 -3567.5
- host_identity_verified  1     0.346 368.03 -3566.5
<none>                                367.69 -3566.4
- property_type           1     0.578 368.27 -3565.1
- number_of_reviews       1     0.602 368.29 -3565.0
- host_response_rate      1     0.916 368.60 -3563.2
- bed_type                1     0.937 368.63 -3563.1
- neighbourhood           1     1.003 368.69 -3562.7
- review_scores_rating    1     1.682 369.37 -3558.9
- amenities               1     1.865 369.55 -3557.9
- instant_bookable        1     2.061 369.75 -3556.8
- cleaning_fee            1     2.136 369.82 -3556.4
```

```
- bathrooms              1     5.429 373.12 -3537.9
- longitude              1     8.408 376.10 -3521.4
- latitude               1    13.115 380.80 -3495.5
- accommodates           1    39.809 407.50 -3354.6
- room_type              1   119.998 487.69 -2980.9

Step:  AIC=-3568.06
log_price ~ accommodates + bathrooms + cleaning_fee + host_response_rate + 
    latitude + longitude + number_of_reviews + review_scores_rating + 
    property_type + room_type + amenities + bed_type + host_has_profile_pic + 
    host_identity_verified + host_since + instant_bookable + 
    neighbourhood

                        Df Sum of Sq    RSS     AIC
- host_since             1     0.073 367.82 -3569.6
- host_has_profile_pic   1     0.163 367.91 -3569.1
- host_identity_verified  1     0.346 368.10 -3568.1
<none>                               367.75 -3568.1
- property_type          1     0.593 368.34 -3566.7
- number_of_reviews      1     0.643 368.39 -3566.4
- host_response_rate     1     0.919 368.67 -3564.9
- bed_type               1     0.932 368.68 -3564.8
- neighbourhood          1     1.005 368.75 -3564.4
- review_scores_rating   1     1.756 369.51 -3560.1
- amenities              1     1.881 369.63 -3559.4
- instant_bookable       1     2.075 369.83 -3558.4
- cleaning_fee           1     2.630 370.38 -3555.2
- bathrooms              1     5.401 373.15 -3539.7
- longitude              1     8.384 376.13 -3523.2
- latitude               1    13.068 380.82 -3497.4
- accommodates           1    39.810 407.56 -3356.3
- room_type              1   120.521 488.27 -2980.4

Step:  AIC=-3569.64
log_price ~ accommodates + bathrooms + cleaning_fee + host_response_rate + 
    latitude + longitude + number_of_reviews + review_scores_rating + 
    property_type + room_type + amenities + bed_type + host_has_profile_pic + 
    host_identity_verified + instant_bookable + neighbourhood

                        Df Sum of Sq    RSS     AIC
- host_has_profile_pic   1     0.168 367.99 -3570.7
```

```
- host_identity_verified  1     0.319 368.14 -3569.8
<none>                                367.82 -3569.6
- property_type           1     0.603 368.43 -3568.2
- number_of_reviews       1     0.630 368.45 -3568.1
- host_response_rate      1     0.904 368.73 -3566.5
- bed_type                1     0.930 368.75 -3566.4
- neighbourhood           1     1.018 368.84 -3565.9
- review_scores_rating    1     1.740 369.56 -3561.8
- amenities               1     1.886 369.71 -3561.0
- instant_bookable        1     2.085 369.91 -3559.9
- cleaning_fee            1     2.633 370.46 -3556.8
- bathrooms               1     5.416 373.24 -3541.2
- longitude               1     8.340 376.16 -3525.0
- latitude                1    13.072 380.90 -3499.0
- accommodates            1    39.805 407.63 -3357.9
- room_type               1   120.614 488.44 -2981.7

Step:  AIC=-3570.69
log_price ~ accommodates + bathrooms + cleaning_fee + host_response_rate + 
    latitude + longitude + number_of_reviews + review_scores_rating + 
    property_type + room_type + amenities + bed_type + host_identity_verified + 
    instant_bookable + neighbourhood

                         Df Sum of Sq    RSS     AIC
- host_identity_verified  1     0.344 368.34 -3570.7
<none>                                367.99 -3570.7
- property_type           1     0.597 368.59 -3569.3
- number_of_reviews       1     0.636 368.63 -3569.1
- host_response_rate      1     0.918 368.91 -3567.5
- bed_type                1     0.936 368.93 -3567.4
- neighbourhood           1     1.024 369.02 -3566.9
- review_scores_rating    1     1.756 369.75 -3562.8
- amenities               1     1.895 369.89 -3562.0
- instant_bookable        1     2.074 370.07 -3561.0
- cleaning_fee            1     2.671 370.66 -3557.6
- bathrooms               1     5.489 373.48 -3541.9
- longitude               1     8.343 376.33 -3526.1
- latitude                1    13.143 381.14 -3499.7
- accommodates            1    39.658 407.65 -3359.8
- room_type               1   120.765 488.76 -2982.4
```

```
Step:  AIC=-3570.75
log_price ~ accommodates + bathrooms + cleaning_fee + host_response_rate + 
    latitude + longitude + number_of_reviews + review_scores_rating + 
    property_type + room_type + amenities + bed_type + instant_bookable + 
    neighbourhood

                       Df Sum of Sq    RSS     AIC
<none>                              368.34 -3570.7
- property_type         1     0.659 368.99 -3569.0
- number_of_reviews     1     0.784 369.12 -3568.3
- bed_type              1     0.872 369.21 -3567.8
- host_response_rate    1     0.997 369.33 -3567.1
- neighbourhood         1     1.024 369.36 -3567.0
- review_scores_rating  1     1.676 370.01 -3563.3
- instant_bookable      1     1.879 370.22 -3562.2
- amenities             1     1.890 370.23 -3562.1
- cleaning_fee          1     2.829 371.16 -3556.8
- bathrooms             1     5.501 373.84 -3541.9
- longitude             1     8.228 376.56 -3526.8
- latitude              1    13.575 381.91 -3497.5
- accommodates          1    39.493 407.83 -3360.9
- room_type             1   120.848 489.18 -2982.6
```

Hide

```
step_boston_mlr
```

```
Call:
lm(formula = log_price ~ accommodates + bathrooms + cleaning_fee + 
    host_response_rate + latitude + longitude + number_of_reviews + 
    review_scores_rating + property_type + room_type + amenities + 
    bed_type + instant_bookable + neighbourhood, data = Training)

Coefficients:
         (Intercept)          accommodates             bathrooms      cleaning_feeTRUE  
           1.201e+00             8.198e-02             1.230e-01            -9.052e-02  
  host_response_rate              latitude             longitude     number_of_reviews  
          -2.029e-01             3.824e+00             2.225e+00            -4.621e-04  
review_scores_rating         property_type             room_type             amenities  
           4.012e-03             2.448e-03            -6.111e-01             1.780e-06  
            bed_type      instant_bookable         neighbourhood  
           4.922e-02            -6.602e-02             1.312e-04  
```

Hide

```
summary(step_boston_mlr)
```

```
Call:
lm(formula = log_price ~ accommodates + bathrooms + cleaning_fee +
    host_response_rate + latitude + longitude + number_of_reviews +
    review_scores_rating + property_type + room_type + amenities +
    bed_type + instant_bookable + neighbourhood, data = Training)

Residuals:
     Min       1Q   Median       3Q      Max
-1.73087 -0.26077 -0.01602  0.26359  2.44244

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           1.201e+00  3.506e+01   0.034  0.97269
accommodates          8.198e-02  5.509e-03  14.880  < 2e-16 ***
bathrooms             1.230e-01  2.216e-02   5.553 3.16e-08 ***
cleaning_feeTRUE     -9.052e-02  2.273e-02  -3.982 7.06e-05 ***
host_response_rate   -2.029e-01  8.582e-02  -2.364  0.01817 *
latitude              3.824e+00  4.384e-01   8.724  < 2e-16 ***
longitude             2.225e+00  3.276e-01   6.792 1.44e-11 ***
number_of_reviews    -4.621e-04  2.204e-04  -2.096  0.03619 *
review_scores_rating  4.012e-03  1.309e-03   3.065  0.00220 **
property_type         2.448e-03  1.274e-03   1.922  0.05474 .
room_type            -6.111e-01  2.348e-02 -26.029  < 2e-16 ***
amenities             1.780e-06  5.470e-07   3.255  0.00115 **
bed_type              4.922e-02  2.226e-02   2.211  0.02716 *
instant_bookable     -6.602e-02  2.034e-02  -3.246  0.00119 **
neighbourhood         1.312e-04  5.476e-05   2.396  0.01667 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4223 on 2065 degrees of freedom
Multiple R-squared:  0.5994,    Adjusted R-squared:  0.5967
F-statistic: 220.7 on 14 and 2065 DF,  p-value: < 2.2e-16
```

Hide

```
boston_mlr_pred <- predict(boston_mlr, Validation)
accuracy(boston_mlr_pred, Validation$log_price)
```

```
                  ME      RMSE       MAE         MPE     MAPE
Test set -0.003546374 0.4447232 0.3303882 -0.8962284 6.893733
```

Step III: Classification

Code

```
[1] "K-nearest neighbor is an algorithm which determines the nearest or similar cases based on measures selected
for the new data case. In this instance we utilized numeric variables provided to us in the data set and created
a rental property and created values for each of the numeric values we chose to be included within our rental pro
perty. The variables \twe chose are the following variables:\n\t·   \tLog_price\n\t·   \tBedrooms\n\t·   \tBeds\n
\t·   \tBathrooms\n\t·   \tAccommodates\n\t·   \tReview scoring rating\n\t·   \tLongitude\n\t·   \tLatitude\n\tTh
ese variables acted as predictors within our model to determine the k-nearest neighbors. Similarity in the model
is defined as the distance metric between two data points (i.e. hamming, Euclidean). However, a difficulty of thi
s model, is trying to determining the correct k-value for the model to predict. We chose 9 for our k-value and \t
this is because it provided the best classification performance. To do this we examined the accuracy of the valid
ation set of data by processing different k-values with an accuracy model. We chose 9 because as you can see from
the results attached below, it has our highest accuracy rate (see screenshot below). By choosing 9, we are maximi
zing \tour data set and not choosing such a high k-value where it doesn’t completely ignore the information from
the predictors."
```

Hide

```
# Part I k-nearest neighbors
boston2020 <-boston1[, c(2, 6, 7, 17, 18, 22, 23, 24, 1, 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 20, 21)]
# Step 2 Partitioned the boston1 data set to 60%/40%
set.seed(220)
rental1 <- sample_n(boston2020, 3468)
bostontrain <-slice(boston2020, 1:2080)
bostonvalid <-slice(boston2020, 2080:3468)
str(bostontrain)
```

```
'data.frame':	2080 obs. of  24 variables:
 $ log_price             : num  4.6 4.68 4.83 4.09 4.96 ...
 $ accommodates          : int  2 2 6 2 2 1 4 3 2 5 ...
 $ bathrooms             : num  2 1 1 1 1 1 1 1 1 1 ...
 $ latitude              : num  42.3 42.3 42.3 42.3 42.4 ...
 $ longitude             : num  -71 -71.1 -71.1 -71.1 -71.1 ...
 $ review_scores_rating  : num  88 96 100 96 80 100 99 100 89 96 ...
 $ bedrooms              : int  1 1 2 1 1 1 2 2 1 2 ...
 $ beds                  : num  1 1 4 1 1 1 2 2 1 2 ...
 $ id                    : int  14648556 4680055 4274462 2278299 16253186 14916417 8442997 2259813 7575345 982304
...
 $ property_type         : Factor w/ 35 levels "Apartment","Bed & Breakfast",..: 30 12 1 18 1 1 18 12 1 1 ...
 $ room_type             : Factor w/ 3 levels "Entire home/apt",..: 2 2 1 2 1 2 2 1 1 1 ...
 $ amenities             : Factor w/ 67122 levels "{\"Air conditioning\",\"Carbon monoxide detector\",\"First aid
kit\",\"Lock on bedroom door\",\"translation mis"| __truncated__,..: 13512 58394 48034 64902 50261 3940 17253 214
97 30934 20451 ...
 $ bed_type              : Factor w/ 5 levels "Airbed","Couch",..: 5 5 5 5 5 5 5 5 5 5 ...
 $ cancellation_policy   : Factor w/ 5 levels "flexible","moderate",..: 3 3 3 1 1 1 1 1 4 1 ...
 $ cleaning_fee          : logi  TRUE TRUE TRUE FALSE FALSE FALSE ...
 $ description           : Factor w/ 73474 levels "         Cozy & clean on the corner of 5th Street and 2nd Ave
nue. 1 room (Queen bed) in 3 bedroom apartment (2"| __truncated__,..: 62216 66393 6863 61398 41717 2057 49914 119
18 46947 8233 ...
 $ host_has_profile_pic  : Factor w/ 3 levels "","f","t": 3 3 3 3 3 3 3 3 3 3 ...
 $ host_identity_verified: Factor w/ 3 levels "","f","t": 3 3 3 2 2 2 3 3 3 3 ...
 $ host_response_rate    : num  1 1 1 1 1 1 1 1 0.88 1 ...
 $ host_since            : Factor w/ 3088 levels "","1/1/11","1/1/12",..: 2442 2078 141 785 1979 2563 468 3004 22
80 1723 ...
 $ instant_bookable      : Factor w/ 2 levels "f","t": 1 2 1 1 2 1 1 1 1 1 ...
 $ name                  : Factor w/ 73350 levels " 1 Bed Apt in Utopic Williamsburg ",..: 20723 51268 493 9707 1
9414 2583 69112 14037 334 43075 ...
 $ neighbourhood         : Factor w/ 620 levels "","16th Street Heights",..: 500 500 500 281 44 359 591 100 503 8
...
 $ number_of_reviews     : int  12 40 5 0 2 1 61 5 9 0 ...
```

Hide

```
# Min & Max of Predictor values
names(bos)
```

```
 [1] "log_price"              "accommodates"           "bathrooms"              "cleaning_fee"
 [5] "host_response_rate"     "latitude"               "longitude"              "number_of_reviews"
 [9] "review_scores_rating"   "property_type"          "room_type"              "amenities"
[13] "bed_type"               "cancellation_policy"    "description"            "host_has_profile_pic"
[17] "host_identity_verified" "host_since"             "instant_bookable"       "name"
[21] "neighbourhood"
```

Hide

```
accommodates <- runif(1, min(bostontrain$accommodates), max(bostontrain$accommodates))
bathrooms <- runif(1, min(bostontrain$bathrooms), max(bostontrain$bathrooms))
bedrooms<- runif(1, min(bostontrain$bedrooms), max(bostontrain$bedrooms))
beds<- runif(1, min(bostontrain$beds), max(bostontrain$beds))
log_price <- runif(1, min(bostontrain$log_price), max(bostontrain$log_price))
review_scores_rating <- runif(1, min(bostontrain$review_scores_rating), max(bostontrain$review_scores_rating))
latitude <- runif(1, min(bostontrain$latitude), max(bostontrain$latitude))
longitude <- runif(1, min(bostontrain$longitude), max(bostontrain$longitude))

names(bostontrain)
```

```
 [1] "log_price"              "accommodates"           "bathrooms"              "latitude"
 [5] "longitude"              "review_scores_rating"   "bedrooms"               "beds"
 [9] "id"                     "property_type"          "room_type"              "amenities"
[13] "bed_type"               "cancellation_policy"    "cleaning_fee"           "description"
[17] "host_has_profile_pic"   "host_identity_verified" "host_response_rate"     "host_since"
[21] "instant_bookable"       "name"                   "neighbourhood"          "number_of_reviews"
```

Hide

```
log_price
```

```
[1] 3.982864
```

Hide

```
accommodates
```

```
[1] 14.47223
```

Hide

```
bathrooms
```

```
[1] 1.777626
```

Hide

```
latitude
```

```
[1] 42.33326
```

Hide

```
longitude
```

```
[1] -70.9895
```

Hide

```
review_scores_rating
```

```
[1] 71.25128
```

Hide

```
bedrooms
```

```
[1] 0.1491482
```

Hide

```
beds
```

```
[1] 12.2405
```

Hide

```
# Creating rental_fee dataframe
colnames(bostontrain)
```

```
 [1] "log_price"              "accommodates"           "bathrooms"              "latitude"
 [5] "longitude"              "review_scores_rating"   "bedrooms"               "beds"
 [9] "id"                     "property_type"          "room_type"              "amenities"
[13] "bed_type"               "cancellation_policy"    "cleaning_fee"           "description"
[17] "host_has_profile_pic"   "host_identity_verified" "host_response_rate"     "host_since"
[21] "instant_bookable"       "name"                   "neighbourhood"          "number_of_reviews"
```

Hide

```
rental_fee <- data.frame(log_price=5.89,
                         accommodates=11.0,
                         bathrooms=1.5,
                         latitude=42.26,
                         longitude=-71.0,
                         review_scores_rating=26.0,
                         bedrooms=3.0,
                         beds=11.0)

train.norm <- bostontrain
valid.norm <- bostonvalid
rental.norm <- rental1

install.packages('caret')
```

```
Error in install.packages : Updating loaded packages
```

Hide

```
library(caret)
norm.values <- preProcess(bostontrain[, 2:8], method=c("center", "scale"))
train.norm[, 2:8] <- predict(norm.values, bostontrain[, 2:8])
valid.norm[, 2:8] <- predict(norm.values, bostonvalid[, 2:8])
rental.norm[, 2:8] <- predict(norm.values, rental1[, 2:8])
new.norm <- predict(norm.values, rental_fee)
# Use Knn Function to find nearest neighbors
install.packages("FNN")
```

```
Installing package into '/Users/josemartinez/Library/R/3.6/library'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/FNN_1.1.3.tgz'
Content type 'application/x-gzip' length 136106 bytes (132 KB)
==================================================
downloaded 132 KB
```

```
The downloaded binary packages are in
    /var/folders/6v/wsr694r57n9dfsdxftfdysbh0000gn/T//RtmplIEjsq/downloaded_packages
```

Hide

```
install.packages("caret")
```

```
Installing package into '/Users/josemartinez/Library/R/3.6/library'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/caret_6.0-86.tgz'
Content type 'application/x-gzip' length 6252869 bytes (6.0 MB)
==================================================
downloaded 6.0 MB
```

```
The downloaded binary packages are in
    /var/folders/6v/wsr694r57n9dfsdxftfdysbh0000gn/T//RtmplIEjsq/downloaded_packages
```

Hide

```
library(FNN)
nn <- knn(train = train.norm[, 2:8], test = new.norm[, 2:8],
          cl=train.norm[, 15], k=9)
nn
```

```
[1] TRUE
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
[1,]  936 2003 1221 1848 1006   52  424  565  170
attr(,"nn.dist")
        [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]
[1,] 6.68265 8.727249 8.970678 9.055821 9.148725 9.170462 9.504666 9.809327 9.900007
Levels: TRUE
```

Hide

```
row.names(bostontrain)[attr(nn, "nn.index")]
```

```
[1] "936"  "2003" "1221" "1848" "1006" "52"   "424"  "565"  "170"
```

Hide

```
# Accuracy
accuracy.rental<- data.frame(k = seq(1, 10, 1), accuracy = rep(0, 10))
for(i in 1:10) {
  knn.pred <- knn(train.norm[, 2:8], valid.norm[, 2:8],
                  cl = train.norm[, 10], k = i)
  accuracy.rental[i, 2] <- confusionMatrix(knn.pred, valid.norm[, 10])$overall[1]
}
```

```
longer object length is not a multiple of shorter object lengthLevels are not in the same order for reference and
data. Refactoring data to match.Levels are not in the same order for reference and data. Refactoring data to matc
h.longer object length is not a multiple of shorter object lengthLevels are not in the same order for reference a
nd data. Refactoring data to match.Levels are not in the same order for reference and data. Refactoring data to m
atch.longer object length is not a multiple of shorter object lengthLevels are not in the same order for referenc
e and data. Refactoring data to match.longer object length is not a multiple of shorter object lengthLevels are n
ot in the same order for reference and data. Refactoring data to match.longer object length is not a multiple of
shorter object lengthLevels are not in the same order for reference and data. Refactoring data to match.longer ob
ject length is not a multiple of shorter object lengthLevels are not in the same order for reference and data. Re
factoring data to match.longer object length is not a multiple of shorter object lengthLevels are not in the same
order for reference and data. Refactoring data to match.longer object length is not a multiple of shorter object
lengthLevels are not in the same order for reference and data. Refactoring data to match.
```

Hide

```
view(accuracy.rental)
```

Part II. Naive Bayes

Code

```
[1] "After generating the bins for prices that would be categorized as ‘student budget’, ‘below average’, ‘above
average’, and ‘pricey dig’, we started to look for variables that would influence the price and rating. It was de
termine that the\tproperty_type, cancellation_policy, bed_type, and cleaning_fee  were among the \tinfluential va
riables that would influence the price and price rating. After slicing, training, and validating our data, the pr
op table gave us an indication of where our probabilities would lie if we selected the price rating/cancellation
policy variable as a sample.\n\n\tWhen we looked at the prediction outcome of our model, we created a fictional a
partment that would fall under the ‘student budget’ category along with a real bed, flexible cancellation policy,
and existing cleaning fee. The model did well in interpreting that we would fall in the student budget category.
To further examine the model, a confusion matrix was created to the test the accuracy for both the training and v
alidation sets. The accuracy table noted above shows that our model was 94.24% accurate. This means that in terms
prediction our model performance was excellent. The training accuracy was slightly above 94% but still within ran
ge of performance for validation."
```

Hide

```
library(e1071)
# Part C
boston2 <- boston1 # copy boston 1 df to preserve orginal data
summary(boston2$log_price) # capture the range
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.833   4.382   4.913   4.884   5.298   7.244 
```

Hide

```
summary(boston2$nightly_price) # capture the nightly price and compare to log
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   17.0    80.0   136.0   165.6   200.0  1400.0 
```

Hide

```
# create bins using the cut function to create 4 price categories
boston2$log_price_rating <- cut(boston2$log_price,
                         breaks=c(2.833, 4.382, 5.298, 7.244, Inf),
                         labels=c("Student Budget","Below Average","Above Average","Pricey Dig"))
summary(boston2$log_price_rating)
```

```
Student Budget  Below Average  Above Average     Pricey Dig 
           827           1739            901              1 
```

Hide

```
# Part D
boston2$log_price <- factor(boston2$log_price)
boston2$property_type <- factor(boston2$property_type)
boston2$cancellation_policy <- factor(boston2$cancellation_policy)
boston2$bed_type <- factor(boston2$bed_type)
boston2$cleaning_fee <- factor(boston2$cleaning_fee)
boston2$log_price_rating <- factor(boston2$log_price_rating)
# create training and validation sets
selected.var <- c(2, 8, 9, 10, 26)
train.index <- sample(c(1:dim(boston2)[1]), dim(boston2)[1]*.60)
boston2_train <- boston2[train.index, selected.var]
boston2_val <- boston2[-train.index, selected.var]
# run naive bayes
boston2.nb <- naiveBayes(log_price_rating ~., data = boston2_train)
boston2.nb
```

```
Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
Student Budget  Below Average  Above Average     Pricey Dig
  0.2471153846   0.4985576923   0.2538461538   0.0004807692

Conditional probabilities:
                log_price
Y                  2.833213344  2.995732274  3.091042453   3.17805383  3.218875825  3.295836866   3.33220451
  Student Budget 0.0000000000 0.0019455253 0.0019455253 0.0019455253 0.0058365759 0.0038910506 0.0058365759
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                   3.36729583  3.401197382  3.433987204  3.465735903  3.496507561  3.526360525  3.555348061
  Student Budget 0.0097276265 0.0214007782 0.0077821012 0.0077821012 0.0097276265 0.0019455253 0.0252918288
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                  3.583518938  3.610917913   3.63758616  3.663561646  3.688879454  3.713572067  3.737669618
  Student Budget 0.0000000000 0.0058365759 0.0097276265 0.0175097276 0.0311284047 0.0097276265 0.0038910506
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                  3.784189634   3.80666249  3.828641396  3.850147602  3.871201011  3.891820298  3.912023005
  Student Budget 0.0077821012 0.0272373541 0.0077821012 0.0097276265 0.0038910506 0.0136186770 0.0836575875
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                  3.931825633  3.951243719  3.970291914  3.988984047  4.007333185  4.025351691  4.043051268
  Student Budget 0.0077821012 0.0038910506 0.0077821012 0.0058365759 0.0603112840 0.0077821012 0.0058365759
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                  4.060443011  4.077537444  4.094344562  4.110873864  4.127134385  4.143134726  4.158883083
  Student Budget 0.0116731518 0.0175097276 0.0778210117 0.0058365759 0.0116731518 0.0175097276 0.0077821012
```

```
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                 4.17438727  4.189654742  4.204692619  4.219507705  4.234106505  4.248495242  4.262679877
  Student Budget 0.0914396887 0.0019455253 0.0194552529 0.0097276265 0.0505836576 0.0642023346 0.0097276265
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                4.276666119  4.290459441  4.304065093  4.317488114   4.33073334  4.343805422  4.356708827
  Student Budget 0.0097276265 0.0019455253 0.0116731518 0.0953307393 0.0058365759 0.0019455253 0.0077821012
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                4.369447852  4.382026635  4.394449155  4.406719247  4.418840608  4.430816799  4.442651256
  Student Budget 0.0330739300 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0327868852 0.0019286403 0.0028929605 0.0019286403 0.0038572806 0.0241080039
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                4.454347296  4.477336814   4.48863637   4.49980967  4.510859507  4.521788577  4.532599493
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0038572806 0.0096432015 0.0154291225 0.0270009643 0.0009643202 0.0009643202 0.0028929605
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                4.543294782  4.553876892  4.564348191  4.574710979  4.584967479   4.59511985  4.605170186
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0009643202 0.0260366442 0.0009643202 0.0048216008 0.0067502411 0.0453230473 0.0520732883
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                4.624972813  4.634728988  4.644390899   4.65396035  4.663439094  4.672828834  4.682131227
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0009643202 0.0028929605 0.0038572806 0.0077145612 0.0009643202 0.0009643202 0.0067502411
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                4.691347882  4.700480366  4.709530201  4.718498871  4.727387819  4.736198448  4.744932128
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0086788814 0.0231436837 0.0019286403 0.0019286403 0.0009643202 0.0028929605 0.0135004822
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                log_price
Y                4.753590191  4.762173935  4.770684624  4.779123493  4.787491743  4.795790546  4.804021045
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0019286403 0.0009643202 0.0019286403 0.0096432015 0.0270009643 0.0009643202 0.0028929605
```

```
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
               log_price
Y                 4.812184355  4.820281566  4.828313737  4.836281907  4.844187086  4.852030264  4.859812404
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0028929605 0.0405014465 0.0019286403 0.0028929605 0.0028929605 0.0395371263
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
               log_price
Y                  4.86753445  4.882801923  4.890349128    4.8978398  4.905274778  4.912654886  4.919980926
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0202507232 0.0019286403 0.0019286403 0.0038572806 0.0192864031 0.0028929605 0.0019286403
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
               log_price
Y                 4.927253685  4.934473933  4.941642423   4.94875989  4.955827058   4.96284463    4.9698133
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0019286403 0.0144648023 0.0163934426 0.0000000000 0.0009643202 0.0009643202 0.0057859209
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
               log_price
Y                 4.976733742  4.983606622  4.990432587  4.997212274  5.003946306  5.010635294  5.017279837
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0096432015 0.0009643202 0.0086788814 0.0038572806 0.0241080039 0.0597878496 0.0009643202
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
               log_price
Y                 5.023880521  5.030437921  5.036952602  5.043425117  5.049856007  5.056245805  5.062595033
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0038572806 0.0009643202 0.0019286403 0.0067502411 0.0009643202 0.0009643202 0.0048216008
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
               log_price
Y                 5.068904202  5.075173815  5.081404365  5.087596335  5.093750201  5.099866428  5.105945474
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0144648023 0.0183220829 0.0000000000 0.0009643202 0.0000000000 0.0009643202 0.0077145612
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
               log_price
Y                 5.111987788  5.117993812  5.123963979  5.129898715  5.135798437  5.141663557  5.147494477
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0019286403 0.0028929605 0.0009643202 0.0135004822 0.0144648023 0.0009643202 0.0028929605
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
               log_price
Y                 5.153291594  5.159055299  5.164785974  5.170483995  5.176149733   5.18178355  5.187385806
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0009643202 0.0048216008 0.0540019286 0.0067502411 0.0028929605 0.0009643202 0.0173577628
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```

```
                 log_price
Y                 5.192956851  5.198497031  5.204006687  5.209486153  5.214935758  5.220355825  5.225746674
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0163934426 0.0028929605 0.0000000000 0.0009643202 0.0038572806 0.0163934426 0.0038572806
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                 log_price
Y                 5.231108617  5.236441963  5.241747015  5.247024072  5.252273428  5.257495372  5.262690189
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0019286403 0.0038572806 0.0173577628 0.0106075217 0.0019286403 0.0009643202 0.0019286403
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
                 log_price
Y                 5.272999559  5.278114659  5.283203729  5.288267031  5.293304825  5.298317367  5.303304908
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0183220829 0.0019286403 0.0000000000 0.0019286403 0.0279652845 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.1212121212 0.0056818182
                 log_price
Y                 5.313205979  5.318119994  5.323009979  5.332718793   5.33753808  5.342334252  5.347107531
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0018939394 0.0056818182 0.0037878788 0.0000000000 0.0113636364 0.0170454545
                 log_price
Y                 5.356586275  5.361292166  5.370638028  5.375278408  5.384495063   5.38907173  5.393627546
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0018939394 0.0075757576 0.0075757576 0.0000000000 0.0113636364 0.0189393939
                 log_price
Y                 5.407171771  5.416100402  5.429345629  5.433722004  5.438079309  5.455321115  5.459585514
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0037878788 0.0492424242 0.0000000000 0.0151515152 0.0075757576 0.0018939394 0.0075757576
                 log_price
Y                 5.463831805  5.468060141  5.476463552  5.480638923  5.501258211  5.505331536  5.509388337
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0037878788 0.0037878788 0.0113636364 0.0075757576 0.0037878788 0.0056818182
                 log_price
Y                 5.517452896  5.521460918  5.525452939  5.541263545  5.549076085  5.556828062  5.560681631
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0265151515 0.0776515152 0.0018939394 0.0056818182 0.0018939394 0.0208333333 0.0113636364
                 log_price
```

```
Y                 5.564520407  5.575949103  5.579729826  5.587248658   5.59471138  5.598421959  5.616771098
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0018939394 0.0018939394 0.0037878788 0.0037878788 0.0018939394 0.0037878788 0.0359848485
                log_price
Y                 5.620400866  5.631211782  5.634789603  5.638354669  5.648974238   5.65248918  5.655991811
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0056818182 0.0056818182 0.0018939394 0.0018939394 0.0056818182 0.0018939394
                log_price
Y                 5.659482216   5.66296048  5.666426688  5.669880923  5.673323267  5.676753802  5.680172609
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0018939394 0.0037878788 0.0037878788 0.0018939394 0.0037878788 0.0000000000
                log_price
Y                 5.683579767  5.686975356  5.697093487  5.700443573  5.703782475  5.720311777  5.723585102
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0018939394 0.0151515152 0.0056818182 0.0208333333 0.0435606061 0.0018939394 0.0018939394
                log_price
Y                 5.726847748  5.739792912  5.752572639  5.758901774  5.765191103  5.768320996  5.780743516
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0018939394 0.0018939394 0.0056818182 0.0018939394 0.0018939394 0.0056818182 0.0037878788
                log_price
Y                 5.783825182  5.789960171  5.793013608  5.796057751  5.799092654  5.802118375  5.814130532
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0132575758 0.0018939394 0.0000000000 0.0037878788 0.0000000000 0.0000000000 0.0037878788
                log_price
Y                  5.82008293  5.826000107  5.831882477  5.834810737  5.843544417   5.84932478  5.855071922
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0018939394 0.0018939394 0.0037878788 0.0018939394 0.0000000000 0.0000000000 0.0094696970
                log_price
Y                 5.857933154  5.863631176  5.877735782  5.883322388  5.886104031  5.888877958  5.894402834
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0435606061 0.0018939394 0.0018939394 0.0018939394 0.0037878788 0.0000000000 0.0000000000
                log_price
Y                 5.908082938  5.910796644  5.926926026  5.937536205  5.940171253  5.948034989  5.963579344
```

```
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0018939394 0.0018939394 0.0075757576 0.0018939394 0.0018939394 0.0000000000 0.0037878788
              log_price
Y                 5.966146739   5.96870756  5.978885765  5.988961417  5.991464547  6.033086222  6.052089169
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0018939394 0.0075757576 0.0151515152 0.0208333333 0.0037878788 0.0056818182
              log_price
Y                 6.061456919  6.084499413  6.109247583   6.11368218  6.163314804  6.171700597  6.173786104
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0075757576 0.0037878788 0.0151515152 0.0018939394 0.0018939394 0.0018939394 0.0018939394
              log_price
Y                 6.184148891  6.204557763  6.212606096  6.214608098  6.222576268  6.224558429  6.226536669
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0018939394 0.0018939394 0.0018939394 0.0170454545 0.0000000000 0.0018939394 0.0018939394
              log_price
Y                 6.242223265  6.251903883  6.261491684  6.263398263  6.284134161  6.308098442  6.309918278
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0037878788 0.0018939394 0.0000000000 0.0018939394 0.0018939394 0.0018939394 0.0018939394
              log_price
Y                 6.324358962  6.326149473  6.342121419   6.34738921  6.354370041  6.388561406  6.391917113
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0000000000 0.0018939394 0.0094696970 0.0018939394 0.0000000000 0.0000000000
              log_price
Y                 6.395261598  6.396929655  6.401917197   6.43775165  6.456769656  6.476972363  6.522092798
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0018939394 0.0000000000 0.0037878788 0.0018939394 0.0056818182 0.0018939394
              log_price
Y                 6.549650742  6.551080335  6.586171655  6.620073207   6.64509097   6.65929392  6.665683718
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0075757576 0.0056818182 0.0000000000 0.0056818182 0.0000000000 0.0018939394 0.0000000000
              log_price
Y                 6.684611728  6.725033642  6.770789424  6.802394763  6.817830571  6.902742737  6.906754779
  Student Budget 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
```

```
  Below Average  0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0113636364 0.0000000000 0.0000000000 0.0018939394 0.0018939394 0.0018939394 0.0037878788
               log_price
Y                 7.13089883  7.150701458  7.244227516
  Student Budget 0.0000000000 0.0000000000 0.0000000000
  Below Average  0.0000000000 0.0000000000 0.0000000000
  Above Average  0.0000000000 0.0018939394 0.0000000000
 [ reached getOption("max.print") -- omitted 1 row ]

               bed_type
Y                     Airbed       Couch       Futon Pull-out Sofa    Real Bed
  Student Budget 0.027237354 0.003891051 0.019455253   0.015564202 0.933852140
  Below Average  0.002892960 0.001928640 0.001928640   0.001928640 0.991321119
  Above Average  0.003787879 0.001893939 0.000000000   0.000000000 0.994318182
  Pricey Dig     0.000000000 0.000000000 0.000000000   0.000000000 1.000000000

               cancellation_policy
Y                   flexible    moderate      strict super_strict_30 super_strict_60
  Student Budget 0.350194553 0.239299611 0.410505837     0.000000000     0.000000000
  Below Average  0.217936355 0.238187078 0.538090646     0.005785921     0.000000000
  Above Average  0.176136364 0.236742424 0.556818182     0.028409091     0.001893939
  Pricey Dig     0.000000000 1.000000000 0.000000000     0.000000000     0.000000000

               cleaning_fee
Y                    FALSE      TRUE
  Student Budget 0.3093385 0.6906615
  Below Average  0.2121504 0.7878496
  Above Average  0.2026515 0.7973485
  Pricey Dig     1.0000000 0.0000000
```

Hide

```
# create a prop table
prop.table(table(boston2_train$cancellation_policy, boston2_train$log_price_rating), margin = 2)
```

```
                 Student Budget Below Average Above Average  Pricey Dig
  flexible            0.350194553   0.217936355   0.176136364 0.000000000
  moderate            0.239299611   0.238187078   0.236742424 1.000000000
  strict              0.410505837   0.538090646   0.556818182 0.000000000
  super_strict_30     0.000000000   0.005785921   0.028409091 0.000000000
  super_strict_60     0.000000000   0.000000000   0.001893939 0.000000000
```

Hide

```
pred.prob <- predict(boston2.nb, newdata = boston2_val, type = "raw")
pred.class <- predict(boston2.nb, newdata = boston2_val)
boston2_df <- data.frame(actual = boston2_val$log_price_rating, predicted = pred.class, pred.prob)
# dummy predictor for apartment type rental with a real bed, above average rating, and flexible cancellation
boston2_df[boston2_val$bed_type == "Real Bed" &
             boston2_val$cancellation_policy == "flexible" &
             boston2_val$cleaning_fee == "TRUE" &
             boston2_val$log_price_rating == "Student Budget",]
```

	actual <fctr>	predicted <fctr>	Student.Budget <dbl>	Below.Average <dbl>	Above.Average <dbl>	Pricey.Dig <dbl>
74	Student Budget	Student Budget	0.7302980	0.19023578	0.079466231	0.00000000107777005
126	Student Budget	Student Budget	0.9748818	0.01771727	0.007400946	0.00000000010037620
172	Student Budget	Student Budget	0.7830988	0.15299243	0.063908753	0.00000000086676994
178	Student Budget	Student Budget	0.8633542	0.09638386	0.040261943	0.00000000054605731
202	Student Budget	Student Budget	0.9748818	0.01771727	0.007400946	0.00000000010037620
205	Student Budget	Student Budget	0.9769703	0.01624414	0.006785584	0.00000000009203028
209	Student Budget	Student Budget	0.9214687	0.05539245	0.023138810	0.00000000031382281
326	Student Budget	Student Budget	0.8186103	0.12794419	0.053445480	0.00000000072486057
331	Student Budget	Student Budget	0.8903916	0.07731289	0.032295522	0.00000000043801179
383	Student Budget	Student Budget	0.9769703	0.01624414	0.006785584	0.00000000009203028

1-10 of 50 rows Previous 1 2 3 4 5 Next

Hide

```
# Assesing model
library(caret)
pred.class <- predict(boston2.nb, newdata = boston2_train)
confusionMatrix(pred.class, boston2_train$log_price_rating)
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Student Budget Below Average Above Average Pricey Dig
  Student Budget            512             6             3          0
  Below Average               2          1030            16          0
  Above Average               0             1           509          0
  Pricey Dig                  0             0             0          1

Overall Statistics

               Accuracy : 0.9865
                 95% CI : (0.9806, 0.991)
    No Information Rate : 0.4986
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9784

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Student Budget Class: Below Average Class: Above Average Class: Pricey Dig
Sensitivity                         0.9961               0.9932               0.9640         1.0000000
Specificity                         0.9943               0.9827               0.9994         1.0000000
Pos Pred Value                      0.9827               0.9828               0.9980         1.0000000
Neg Pred Value                      0.9987               0.9932               0.9879         1.0000000
Prevalence                          0.2471               0.4986               0.2538         0.0004808
Detection Rate                      0.2462               0.4952               0.2447         0.0004808
Detection Prevalence                0.2505               0.5038               0.2452         0.0004808
Balanced Accuracy                   0.9952               0.9880               0.9817         1.0000000
```

Hide

```
pred.class <- predict(boston2.nb, newdata = boston2_val)
confusionMatrix(pred.class, boston2_val$log_price_rating)
```

```
Confusion Matrix and Statistics

                Reference
Prediction       Student Budget Below Average Above Average Pricey Dig
  Student Budget            306             1            11          0
  Below Average               7           701            47          0
  Above Average               0             0           315          0
  Pricey Dig                  0             0             0          0

Overall Statistics

               Accuracy : 0.9524
                 95% CI : (0.9399, 0.963)
    No Information Rate : 0.5058
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9223

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Student Budget Class: Below Average Class: Above Average Class: Pricey Dig
Sensitivity                         0.9776               0.9986               0.8445                NA
Specificity                         0.9888               0.9213               1.0000                 1
Pos Pred Value                      0.9623               0.9285               1.0000                NA
Neg Pred Value                      0.9935               0.9984               0.9459                NA
Prevalence                          0.2255               0.5058               0.2687                 0
Detection Rate                      0.2205               0.5050               0.2269                 0
Detection Prevalence                0.2291               0.5439               0.2269                 0
Balanced Accuracy                   0.9832               0.9599               0.9223                NA
```

Part III. Classification Tree

Code

```
[1] "The process of building a Classification Tree that predicts the outcome of the cancelation policy of an AirB
nb rental listing involved several steps. After importing and filtering the data into a Boston-only dataframe, we
converted listings from the dataset where the cancelation policy was either super_strict_60 or super_strict_30 in
to just “strict”. Since the assignment was to predict the outcome cancelation policy into only one of three bucke
ts (flexible, moderate, or strict) these other cancelation policies needed to be removed. Luckily there were very
few observations within the Boston dataset that had the super_strict policy. The next step was to replace NULL va
lues in the dataset with NA, and then impute median values to replace the NA values. This ensured that our calcul
ations would function as intended, since the packages we were utilizing cannot handle NULL or NA values in their
computations. Next, we renamed the row names as the ID field of the AIrBnb listing, and removed columns that we d
id not intend to use as a part of the classification tree. We chose the rows to remove by considering the value t
hat they would off to the model, while also contemplating the computational strain that certain categorical varia
bles would cause to our local machines. While the rpart() and rpartplot() packages can handle categorical variabl
es, ones with many different values across the observations in the dataset are computationally cumbersome and the
resulting number of splits in the tree can be unwieldy. We ultimately chose to include 12 input variables, most o
f which were numeric, and a few more manageable categorical values like “Bed_Type” where there were only a few po
ssible values. After that, we confirmed the 12 remianing variables were formatted in the datatype we desired and
then partitioned out data into training and testing datasets.\n\nOnce the training data was ready, we utilized th
e rpart() package with a complexity parameter of 0 to build our initial tree. By setting the CP = 0, we were ensu
ring that rpart() would build a massive tree that would surely overfit the training data. While this may seem lik
e a wasted step since we knew we would not utilize the resulting tree, by doing this we were able to run the prin
tcp() function and find the ideal number of splits for our tree where we would minimize the cross-validation erro
r (denoted as xerror in the console). We then plotted the CP = 0 tree to confirm our suspicions that the tree was
in fact too large. After plotting the tree and finding the CP value that corresponded to the lowest xerror, we re
ran our rpart() function with the xerror minimizing CP value, which resulted in a new classification tree with an
ideal number of splits. For our dataset, the CP value that corresponded to the lowest xerror was 0.00501. \n"
```

Hide

```
library(reshape2)
```

```
Attaching package: ‘reshape2’

The following objects are masked from ‘package:reshape’:

    colsplit, melt, recast

The following object is masked from ‘package:tidyr’:

    smiths
```

Hide

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

head(df)
```

	id <int>	log_price <dbl>	property_type <fctr>	room_type <fctr>
1	6901257	5.010635	Apartment	Entire home/apt
2	6304928	5.129899	Apartment	Entire home/apt
3	7919400	4.976734	Apartment	Entire home/apt
4	13418779	6.620073	House	Entire home/apt
5	3808709	4.744932	Apartment	Entire home/apt
6	12422935	4.442651	Apartment	Private room

6 rows | 1-5 of 29 columns

Hide

```
set.seed(200)
boston3 <- filter(df, city=="Boston")
# Convert 'super_strict_60' & 'super_strict_30' to just 'strict'
boston3 <- data.frame(lapply(boston3, function(x) {gsub("super_", "", x) }))
boston3 <- data.frame(lapply(boston3, function(x) {gsub("_30", "", x) }))
boston3 <- data.frame(lapply(boston3, function(x) {gsub("_60", "", x) }))
# check for 'NA' values
sum(is.na(boston3))
```

```
[1] 659
```

Hide

```
colSums(is.na(boston3))
```

```
                   id            log_price         property_type              room_type 
                    0                    0                     0                      0 
            amenities         accommodates             bathrooms               bed_type 
                    0                    0                     6                      0 
  cancellation_policy         cleaning_fee                  city            description 
                    0                    0                     0                      0 
         first_review host_has_profile_pic host_identity_verified     host_response_rate 
                    0                    0                     0                      0 
           host_since     instant_bookable           last_review               latitude 
                    0                    0                     0                      0 
            longitude                 name         neighbourhood      number_of_reviews 
                    0                    0                     0                      0 
 review_scores_rating        thumbnail_url               zipcode               bedrooms 
                  648                    0                     0                      3 
                 beds 
                    2 
```

Hide

```
# Convert null values to 'NA'
boston3[boston3== ""] <-NA
# Impute median values when 'NA' & convert host_response_rate to numeric
boston3$review_scores_rating <- as.numeric(boston3$review_scores_rating)
boston3$review_scores_rating[is.na(boston3$review_scores_rating)] <- median(boston3$review_scores_rating, na.rm =
TRUE)
boston3$host_response_rate <- as.numeric(sub("%","",boston3$host_response_rate))/100
boston3$host_response_rate[is.na(boston3$host_response_rate)] <- median(boston3$host_response_rate, na.rm=TRUE)
boston3$beds <- as.numeric(boston3$beds)
boston3$beds[is.na(boston3$beds)] <- median(boston3$beds, na.rm = T)
boston3$bathrooms <- as.numeric(boston3$bathrooms)
boston3$bathrooms[is.na(boston3$bathrooms)] <- median(boston3$bathrooms, na.rm=T)
boston3$bedrooms <-as.numeric(boston3$bedrooms)
boston3$bedrooms[is.na(boston3$bedrooms)] <- median(boston3$bedrooms, na.rm=T)

colSums(is.na(boston3))
```

```
                   id             log_price         property_type             room_type 
                    0                     0                     0                     0 
            amenities          accommodates             bathrooms              bed_type 
                    0                     0                     0                     0 
  cancellation_policy          cleaning_fee                  city           description 
                    0                     0                     0                     0 
         first_review host_has_profile_pic host_identity_verified    host_response_rate 
                  621                     0                     0                     0 
           host_since      instant_bookable           last_review              latitude 
                    0                     0                   621                     0 
            longitude                  name          neighbourhood     number_of_reviews 
                    0                     0                     0                     0 
 review_scores_rating         thumbnail_url               zipcode              bedrooms 
                    0                   134                    26                     0 
                 beds 
                    0 
```

Hide

```
sum(is.na(boston3))
```

```
[1] 1402
```

Hide

```
# rename rows as id field
rownames(boston3) <- boston3[,1]
# remove fields we do not intent to use
names(boston3)
```

```
 [1] "id"                     "log_price"              "property_type"          "room_type"
 [5] "amenities"              "accommodates"           "bathrooms"              "bed_type"
 [9] "cancellation_policy"    "cleaning_fee"           "city"                   "description"
[13] "first_review"           "host_has_profile_pic"   "host_identity_verified" "host_response_rate"
[17] "host_since"             "instant_bookable"       "last_review"            "latitude"
[21] "longitude"              "name"                   "neighbourhood"          "number_of_reviews"
[25] "review_scores_rating"   "thumbnail_url"          "zipcode"                "bedrooms"
[29] "beds"
```

Hide

```
boston3 <- boston3[-c(1, 3, 5, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 26, 27)]
# confirm numeric variables are in fact numeric, convert if not
class(boston3$log_price)
```

```
[1] "factor"
```

Hide

```
boston3$log_price <- as.numeric((boston3$log_price))
class(boston3$room_type) #factor
```

```
[1] "factor"
```

Hide

```
class(boston3$accommodates)
```

```
[1] "factor"
```

Hide

```
boston3$accommodates <- as.numeric(boston3$accommodates)
class(boston3$bathrooms)
```

```
[1] "numeric"
```

Hide

```
class(boston3$bed_type) #factor
```

```
[1] "factor"
```

Hide

```
class(boston3$cancellation_policy) #factor
```

```
[1] "factor"
```

Hide

```
class(boston3$cleaning_fee) #factor
```

```
[1] "factor"
```

Hide

```
class(boston3$instant_bookable) #factor
```

```
[1] "factor"
```

Hide

```
class(boston3$number_of_reviews)
```

```
[1] "factor"
```

Hide

```
boston3$number_of_reviews <- as.numeric(boston3$number_of_reviews)
class(boston3$review_scores_rating)
```

```
[1] "numeric"
```

Hide

```
class(boston3$bedrooms)
```

```
[1] "numeric"
```

Hide

```
class(boston3$beds)
```

```
[1] "numeric"
```

Hide

```
# Create data partition of dataset
train.index <- createDataPartition(boston3$cancellation_policy,
                                   p = 0.60,  #percentage split, enter desired portion for training data (60/40 s
plit)
                                   list = FALSE,  #tells it that we do not want it to come out as a list
                                   times = 1)

bos3_train <- boston3[train.index ,]
bos3_valid <- boston3[-train.index ,]

names(bos3_train)
```

```
 [1] "log_price"            "room_type"            "accommodates"         "bathrooms"
 [5] "bed_type"             "cancellation_policy"  "cleaning_fee"         "instant_bookable"
 [9] "number_of_reviews"    "review_scores_rating" "bedrooms"             "beds"
```

Hide

```
names(bos3_valid)
```

```
 [1] "log_price"            "room_type"            "accommodates"         "bathrooms"
 [5] "bed_type"             "cancellation_policy"  "cleaning_fee"         "instant_bookable"
 [9] "number_of_reviews"    "review_scores_rating" "bedrooms"             "beds"
```

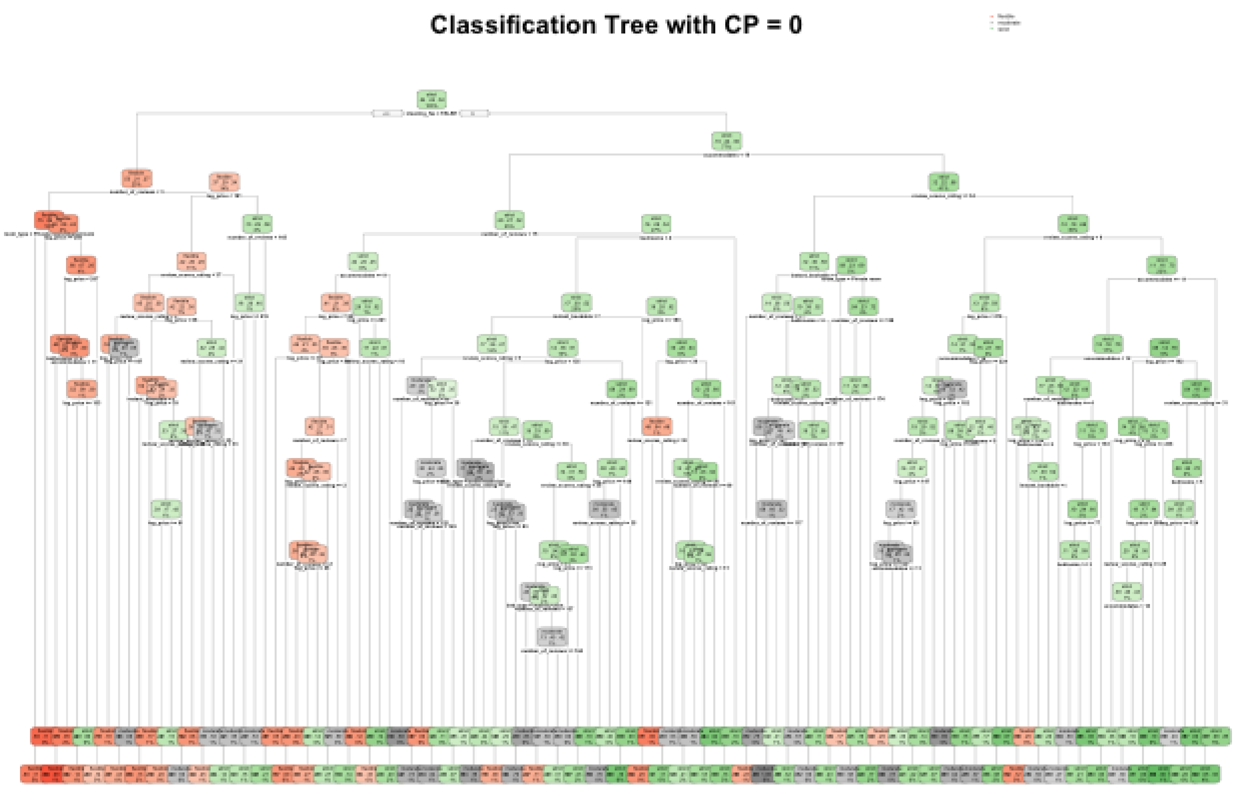
Hide

```
# CREATING CLASSIFICATION TREES WITH VARIOUS COMPLEXITY PARAMETERS
options(scipen = 999)
# rpart with cp = 0 --- creates unpruned very large tree
tree_bos3_cp0 <- rpart(cancellation_policy ~ .,
                       data=bos3_train,
                       method = "class",
                       xval = 5,
                       cp = 0)
printcp(tree_bos3_cp0)
```

```
Classification tree:
rpart(formula = cancellation_policy ~ ., data = bos3_train, method = "class", 
    xval = 5, cp = 0)

Variables actually used in tree construction:
 [1] accommodates         bathrooms            bed_type             bedrooms             cleaning_fee        
 [6] instant_bookable     log_price            number_of_reviews    review_scores_rating room_type           

Root node error: 998/2083 = 0.47912

n= 2083 

          CP nsplit rel error  xerror     xstd
1  0.1272545      0   1.00000 1.00000 0.022846
2  0.0115230      1   0.87275 0.87275 0.022557
3  0.0100200      3   0.84970 0.86874 0.022542
4  0.0046760      4   0.83968 0.85872 0.022504
5  0.0035070      7   0.82565 0.87375 0.022561
6  0.0027555      9   0.81864 0.88878 0.022613
7  0.0025050     14   0.80461 0.89579 0.022635
8  0.0020040     29   0.75150 0.90581 0.022666
9  0.0017307     44   0.71944 0.91283 0.022685
10 0.0015030     60   0.68838 0.90982 0.022677
11 0.0013360     66   0.67936 0.90281 0.022657
12 0.0010020     72   0.67134 0.92285 0.022712
13 0.0006680     89   0.65331 0.93587 0.022744
14 0.0005010     98   0.64729 0.94890 0.022771
15 0.0002505    104   0.64429 0.95892 0.022790
16 0.0002004    108   0.64329 0.96794 0.022806
17 0.0000000    113   0.64228 0.96794 0.022806
```

Hide

```
rpart.plot(tree_bos3_cp0, 
           main = "Classification Tree with CP = 0")
```



Hide

```
# results before pruning
cp0 <- printcp(tree_bos3_cp0)
```

```
Classification tree:
rpart(formula = cancellation_policy ~ ., data = bos3_train, method = "class",
    xval = 5, cp = 0)

Variables actually used in tree construction:
 [1] accommodates         bathrooms            bed_type             bedrooms             cleaning_fee
 [6] instant_bookable     log_price            number_of_reviews    review_scores_rating room_type

Root node error: 998/2083 = 0.47912

n= 2083

          CP nsplit rel error  xerror     xstd
1  0.1272545      0   1.00000 1.00000 0.022846
2  0.0115230      1   0.87275 0.87275 0.022557
3  0.0100200      3   0.84970 0.86874 0.022542
4  0.0046760      4   0.83968 0.85872 0.022504
5  0.0035070      7   0.82565 0.87375 0.022561
6  0.0027555      9   0.81864 0.88878 0.022613
7  0.0025050     14   0.80461 0.89579 0.022635
8  0.0020040     29   0.75150 0.90581 0.022666
9  0.0017307     44   0.71944 0.91283 0.022685
10 0.0015030     60   0.68838 0.90982 0.022677
11 0.0013360     66   0.67936 0.90281 0.022657
12 0.0010020     72   0.67134 0.92285 0.022712
13 0.0006680     89   0.65331 0.93587 0.022744
14 0.0005010     98   0.64729 0.94890 0.022771
15 0.0002505    104   0.64429 0.95892 0.022790
16 0.0002004    108   0.64329 0.96794 0.022806
17 0.0000000    113   0.64228 0.96794 0.022806
```

Hide

```
class(cp0)
```

```
[1] "matrix"
```

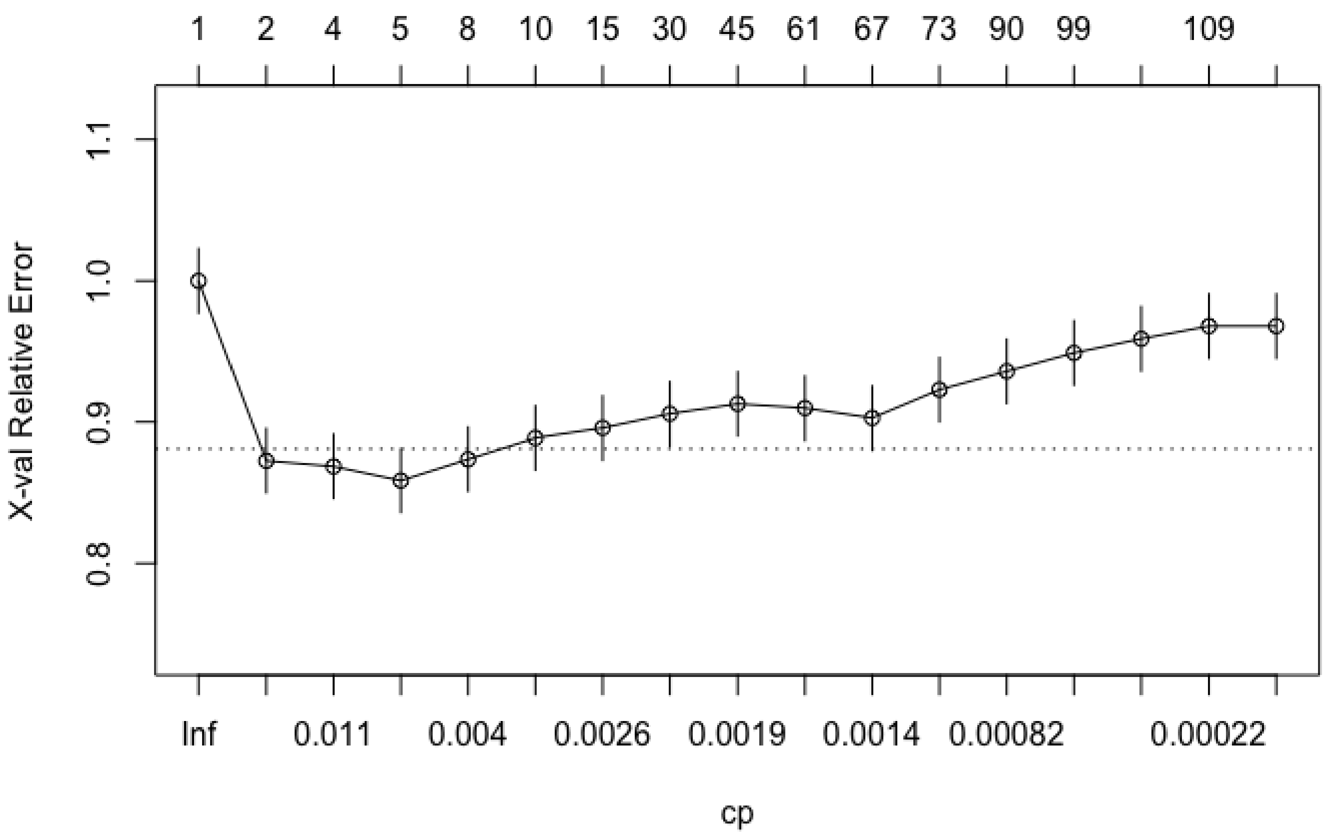
Hide

```
cp0 <- data.frame(cp0)
which.min(cp0$xerror)
```

```
[1] 4
```

Hide

```
plotcp(tree_bos3_cp0)
```



Hide

```
tree_cp0_pred <- predict(tree_bos3_cp0, bos3_train, type = "class")
confusionMatrix(tree_cp0_pred,bos3_train$cancellation_policy)
```

```
Confusion Matrix and Statistics

          Reference
Prediction flexible moderate strict
  flexible      325       68    106
  moderate       45      246    108
  strict        137      177    871

Overall Statistics

               Accuracy : 0.6923
                 95% CI : (0.6719, 0.712)
    No Information Rate : 0.5209
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.4873

 Mcnemar's Test P-Value : 0.0000131

Statistics by Class:

                     Class: flexible Class: moderate Class: strict
Sensitivity                   0.6410          0.5010        0.8028
Specificity                   0.8896          0.9039        0.6854
Pos Pred Value                0.6513          0.6165        0.7350
Neg Pred Value                0.8851          0.8545        0.7617
Prevalence                    0.2434          0.2357        0.5209
Detection Rate                0.1560          0.1181        0.4181
Detection Prevalence          0.2396          0.1916        0.5689
Balanced Accuracy             0.7653          0.7025        0.7441
```

Hide

```
tree_cp0_pred2 <- predict(tree_bos3_cp0, bos3_valid, type = "class")
confusionMatrix(tree_cp0_pred2,bos3_valid$cancellation_policy)
```

```
Confusion Matrix and Statistics

          Reference
Prediction flexible moderate strict
  flexible      170       72    106
  moderate       55       81    131
  strict        112      173    485

Overall Statistics

               Accuracy : 0.5314
                 95% CI : (0.5047, 0.558)
    No Information Rate : 0.5213
    P-Value [Acc > NIR] : 0.23393

                  Kappa : 0.2238

 Mcnemar's Test P-Value : 0.04124

Statistics by Class:

                     Class: flexible Class: moderate Class: strict
Sensitivity                   0.5045         0.24847        0.6717
Specificity                   0.8302         0.82436        0.5701
Pos Pred Value                0.4885         0.30337        0.6299
Neg Pred Value                0.8390         0.78086        0.6146
Prevalence                    0.2433         0.23538        0.5213
Detection Rate                0.1227         0.05848        0.3502
Detection Prevalence          0.2513         0.19278        0.5560
Balanced Accuracy             0.6673         0.53641        0.6209
```

Hide

```
# rpart with xerror minimizing cp value
tree_bos3_cp_min_error <- rpart(cancellation_policy ~ .,
                                data=bos3_train,
                                method = "class",
                                cp = 0.0050100) # complexity parameter that corresponds to the value where the xe
rror was minimized (4th record in the cp0 df, nsplit = 7)
printcp(tree_bos3_cp_min_error)
```

```
Classification tree:
rpart(formula = cancellation_policy ~ ., data = bos3_train, method = "class",
    cp = 0.00501)

Variables actually used in tree construction:
[1] cleaning_fee         log_price            number_of_reviews    review_scores_rating

Root node error: 998/2083 = 0.47912

n= 2083

        CP nsplit rel error  xerror     xstd
1 0.127255      0   1.00000 1.00000 0.022846
2 0.011523      1   0.87275 0.87275 0.022557
3 0.010020      3   0.84970 0.87074 0.022550
4 0.005010      4   0.83968 0.85872 0.022504
```

Hide

```
# results after pruning
cp_min_error.pred <- predict(tree_bos3_cp_min_error, bos3_train, type = "class")
confusionMatrix(cp_min_error.pred, bos3_train$cancellation_policy)
```

```
Confusion Matrix and Statistics

          Reference
Prediction flexible moderate strict
  flexible      244       71     93
  moderate        6       16      7
  strict        257      404    985

Overall Statistics

               Accuracy : 0.5977
                 95% CI : (0.5763, 0.6188)
    No Information Rate : 0.5209
    P-Value [Acc > NIR] : 0.000000000001078

                  Kappa : 0.2514

 Mcnemar's Test P-Value : < 0.00000000000000022

Statistics by Class:

                     Class: flexible Class: moderate Class: strict
Sensitivity                   0.4813        0.032587        0.9078
Specificity                   0.8959        0.991834        0.3377
Pos Pred Value                0.5980        0.551724        0.5984
Neg Pred Value                0.8430        0.768744        0.7712
Prevalence                    0.2434        0.235718        0.5209
Detection Rate                0.1171        0.007681        0.4729
Detection Prevalence          0.1959        0.013922        0.7902
Balanced Accuracy             0.6886        0.512210        0.6228
```

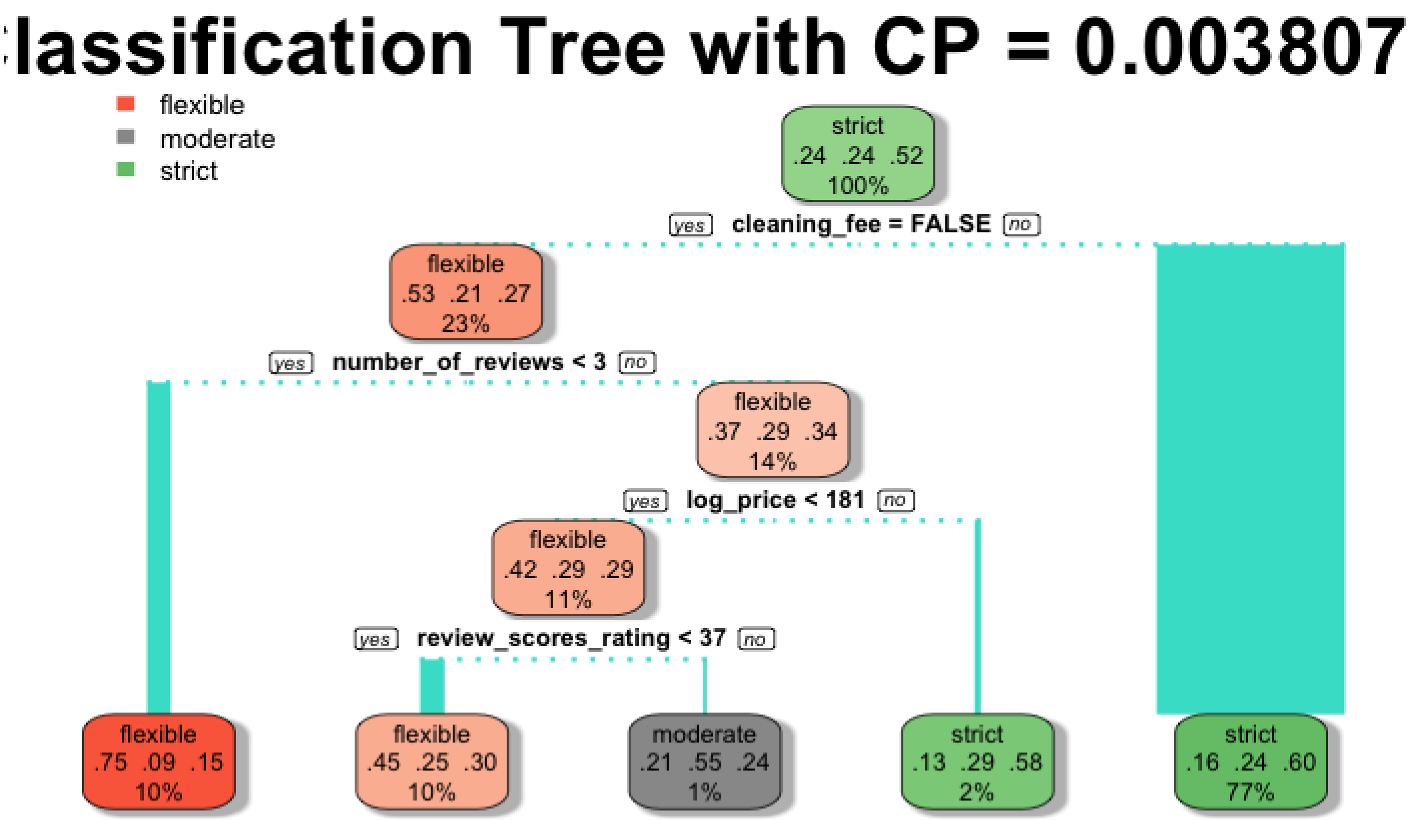
Hide

```
cp_min_error_pred2 <- predict(tree_bos3_cp_min_error, bos3_valid, type = "class")
confusionMatrix(cp_min_error_pred2, bos3_valid$cancellation_policy)
```

```
Confusion Matrix and Statistics

          Reference
Prediction flexible moderate strict
  flexible      160       47     70
  moderate        9       10      3
  strict        168      269    649

Overall Statistics

               Accuracy : 0.5913
                 95% CI : (0.5649, 0.6174)
    No Information Rate : 0.5213
    P-Value [Acc > NIR] : 0.00000009496

                  Kappa : 0.2416

 Mcnemar's Test P-Value : < 0.00000000000000022

Statistics by Class:

                     Class: flexible Class: moderate Class: strict
Sensitivity                   0.4748         0.03067        0.8989
Specificity                   0.8884         0.98867        0.3409
Pos Pred Value                0.5776         0.45455        0.5976
Neg Pred Value                0.8403         0.76816        0.7559
Prevalence                    0.2433         0.23538        0.5213
Detection Rate                0.1155         0.00722        0.4686
Detection Prevalence          0.2000         0.01588        0.7841
Balanced Accuracy             0.6816         0.50967        0.6199
```

Hide

```
# rpart.plot visualization of pruned tree
rpart.plot(tree_bos3_cp_min_error,
           main = "Classification Tree with CP = 0.0038076",
           clip.right.labs = FALSE,
           type = 2,
           branch = .75,
           yesno = 2,
           under = FALSE,
           cex.main = 2.5,
           shadow.col = "gray",
           branch.col = "turquoise",
           branch.lwd = 3,
           branch.lty = 3,
           branch.type = 5,
           gap = 0)
```



Step IV: Clustering

Hide

```
library(cluster)    # clustering algorithms
library(gridExtra)
```

```
Attaching package: ‘gridExtra’

The following object is masked from ‘package:dplyr’:

    combine
```

Hide

```
library(cluster)    # clustering algorithms
library(factoextra) # clustering algorithms & visualization
```

```
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

Hide

```
Data <- boston1
"Converted the categorical variable ‘cancellation_policy’ to numeric with a scale of 1 (flexible) to 5 (super_str
ict_60)."
```

```
[1] "Converted the categorical variable ‘cancellation_policy’ to numeric with a scale of 1 (flexible) to 5 (super
_strict_60)."
```

Hide

```
levels(Data$cancellation_policy)
```

```
[1] "flexible"        "moderate"        "strict"          "super_strict_30" "super_strict_60"
```

Hide

```
# Converting cancellation_policy to numeric
Data$cancellation_policy<-revalue(Data$cancellation_policy,c("flexible"=1,"moderate"=2,"strict"=3,
                                                              "super_strict_30"=4, "super_strict_60"=5))
"Excluded the ‘Cambridge’ and ‘Somerville’ neighborhoods as there were only 5 properties in total."
```

```
[1] "Excluded the ‘Cambridge’ and ‘Somerville’ neighborhoods as there were only 5 properties in total."
```

Hide

```
Data <- filter(Data, neighbourhood!= "Cambridge")
Data <- filter(Data, neighbourhood!= "Somerville")
# Adding nightly price per person
Data <- Data%>%
  mutate(price_per_person = nightly_price/accommodates)
# Remove non-numeric columns
colnames(Data)
```

```
 [1] "id"                     "log_price"              "property_type"          "room_type"
 [5] "amenities"              "accommodates"           "bathrooms"              "bed_type"
 [9] "cancellation_policy"    "cleaning_fee"           "description"            "host_has_profile_pic"
[13] "host_identity_verified" "host_response_rate"     "host_since"             "instant_bookable"
[17] "latitude"               "longitude"              "name"                   "neighbourhood"
[21] "number_of_reviews"      "review_scores_rating"   "bedrooms"               "beds"
[25] "nightly_price"          "price_per_person"
```

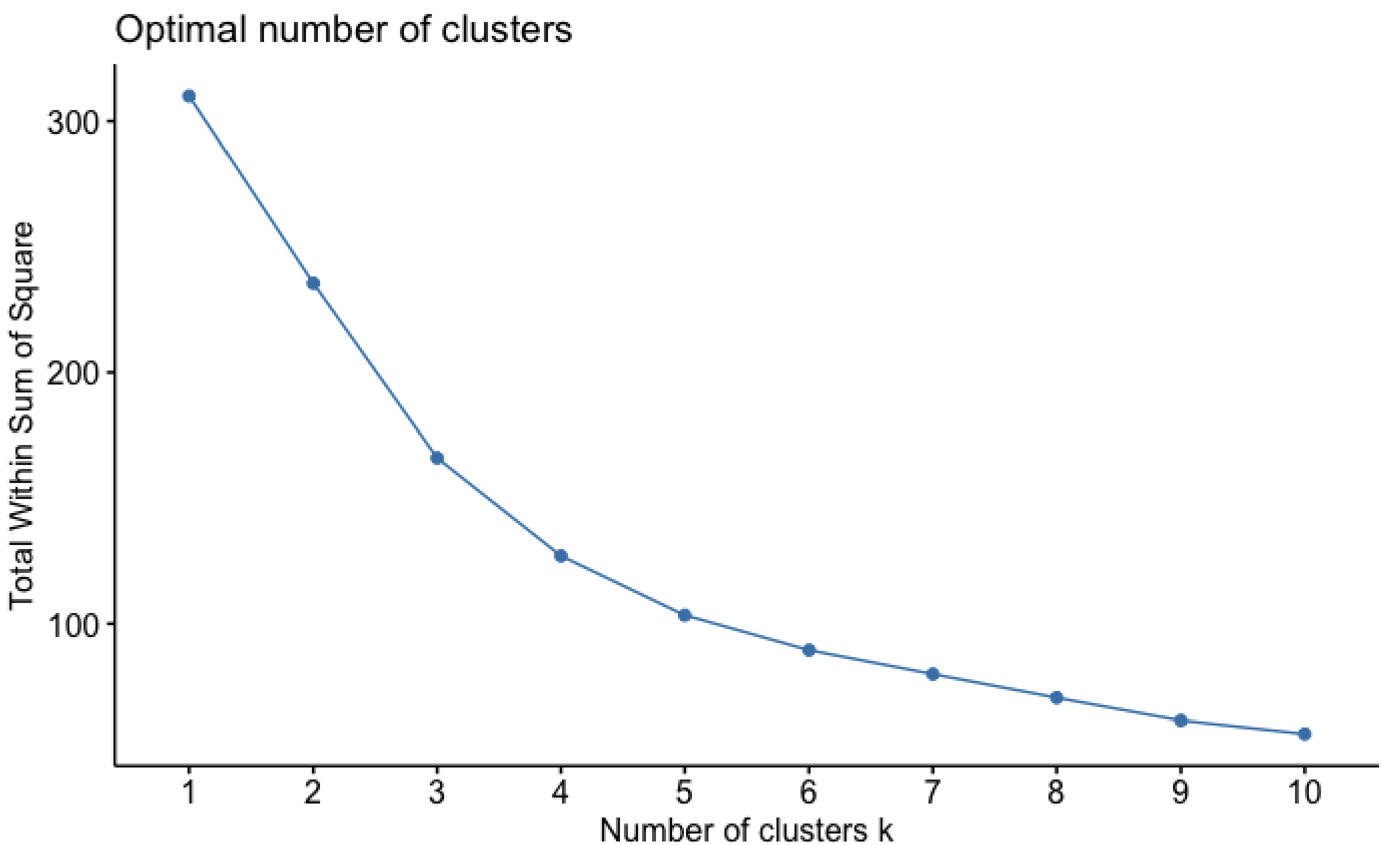
Hide

```
Data<-Data[,-c(1,3:5,8,10:19)]
colnames(Data)
```

```
 [1] "log_price"            "accommodates"         "bathrooms"            "cancellation_policy"
 [5] "neighbourhood"        "number_of_reviews"    "review_scores_rating" "bedrooms"
 [9] "beds"                 "nightly_price"        "price_per_person"
```

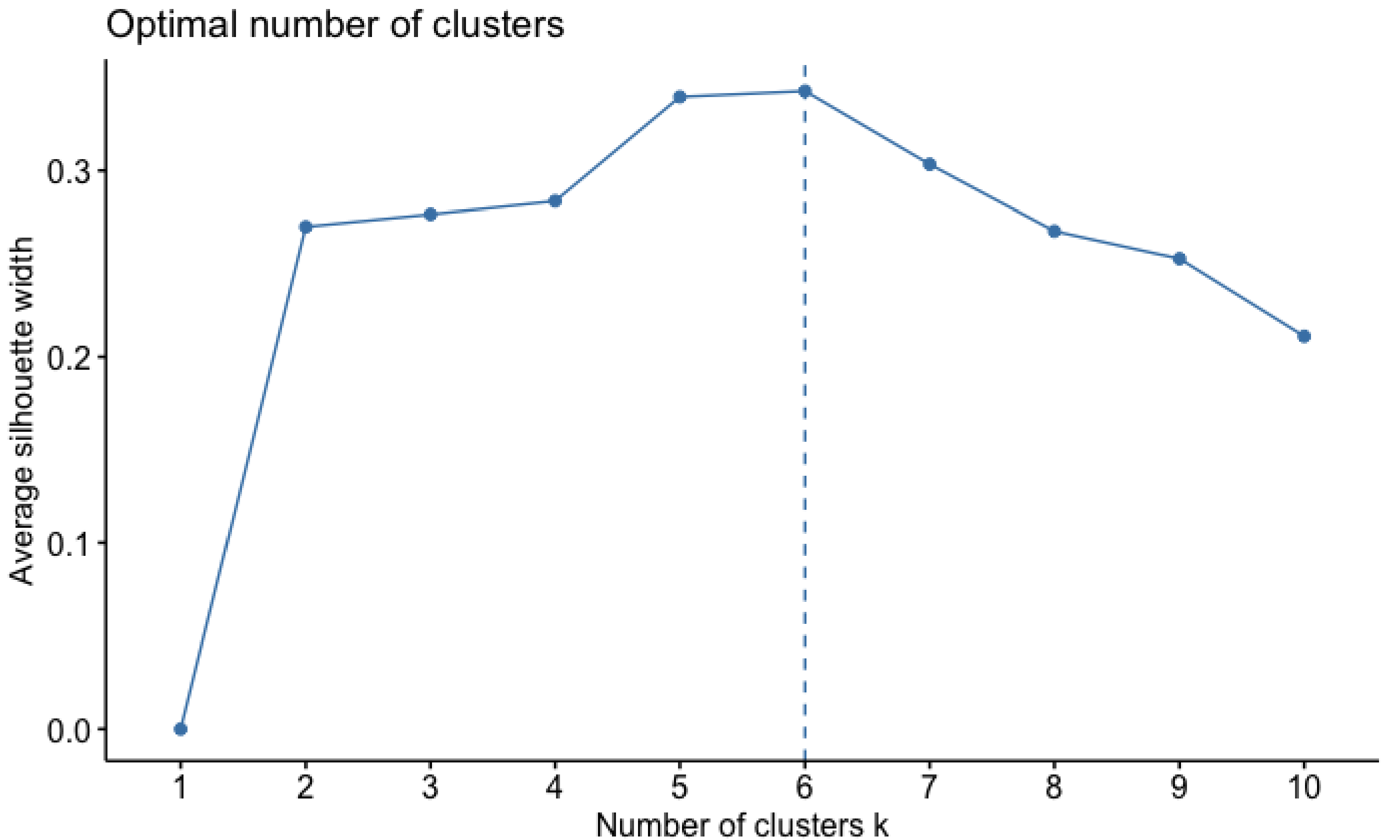
Hide

```
Aggregate_Data<-aggregate(cbind(log_price,accommodates,bathrooms,cancellation_policy, number_of_reviews,review_sc
ores_rating,bedrooms,beds,nightly_price,price_per_person)~neighbourhood,data=Data,mean)

Boston<-data.frame(Aggregate_Data[,-1],row.names=Aggregate_Data$neighbourhood)
Scaled_Data <- scale(Boston)

# Optimal No. of Clusters
# elbow method
set.seed(123)
fviz_nbclust(Scaled_Data, kmeans, method = "wss")
```



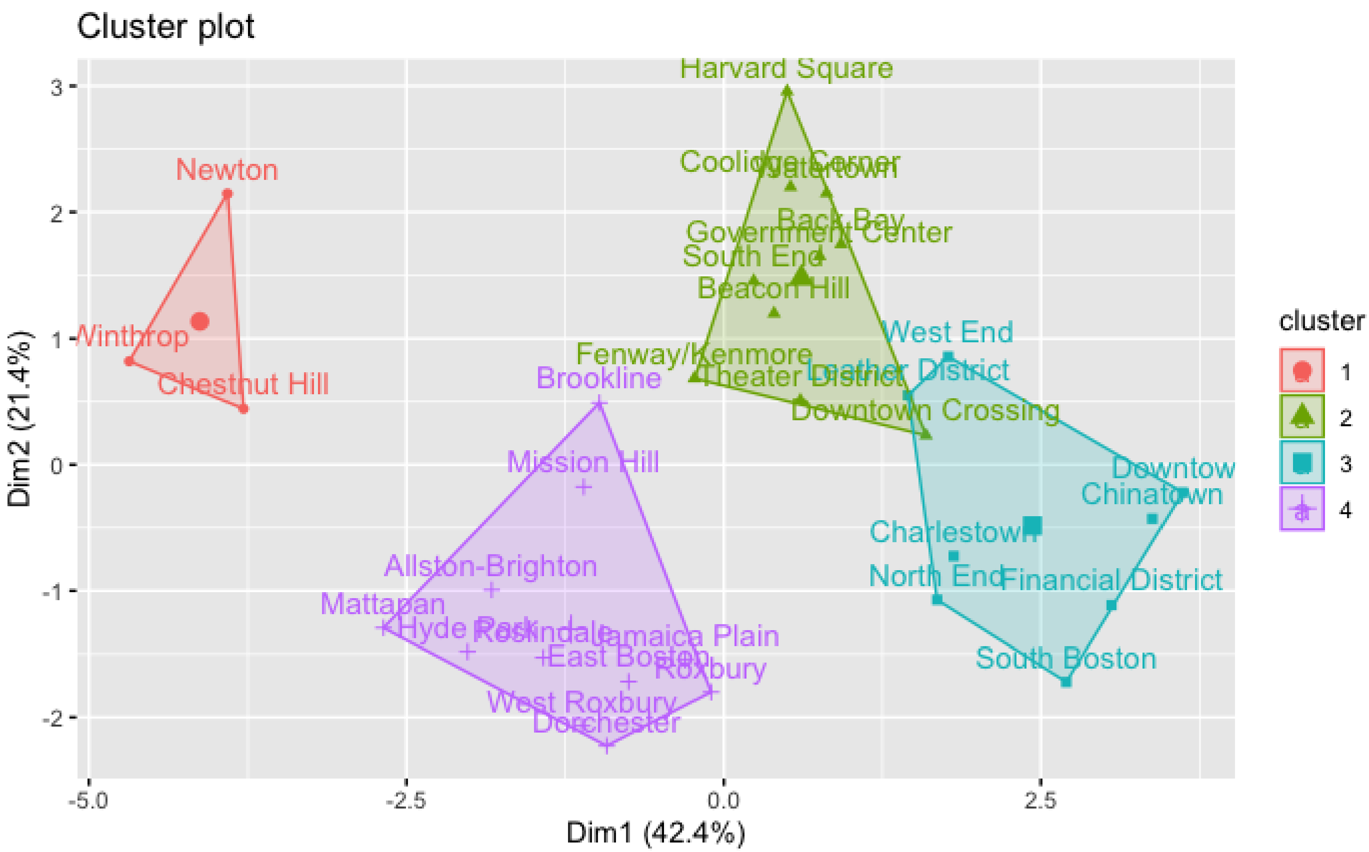
Hide

```
# avg silhouette method
fviz_nbclust(Scaled_Data, kmeans, method = "silhouette")
```



Hide

```
# K-Means Algorithm
set.seed(123)
k4 <- kmeans(Scaled_Data, 4, nstart = 25)
p1 <- fviz_cluster(k4, data = Scaled_Data)
p1
```



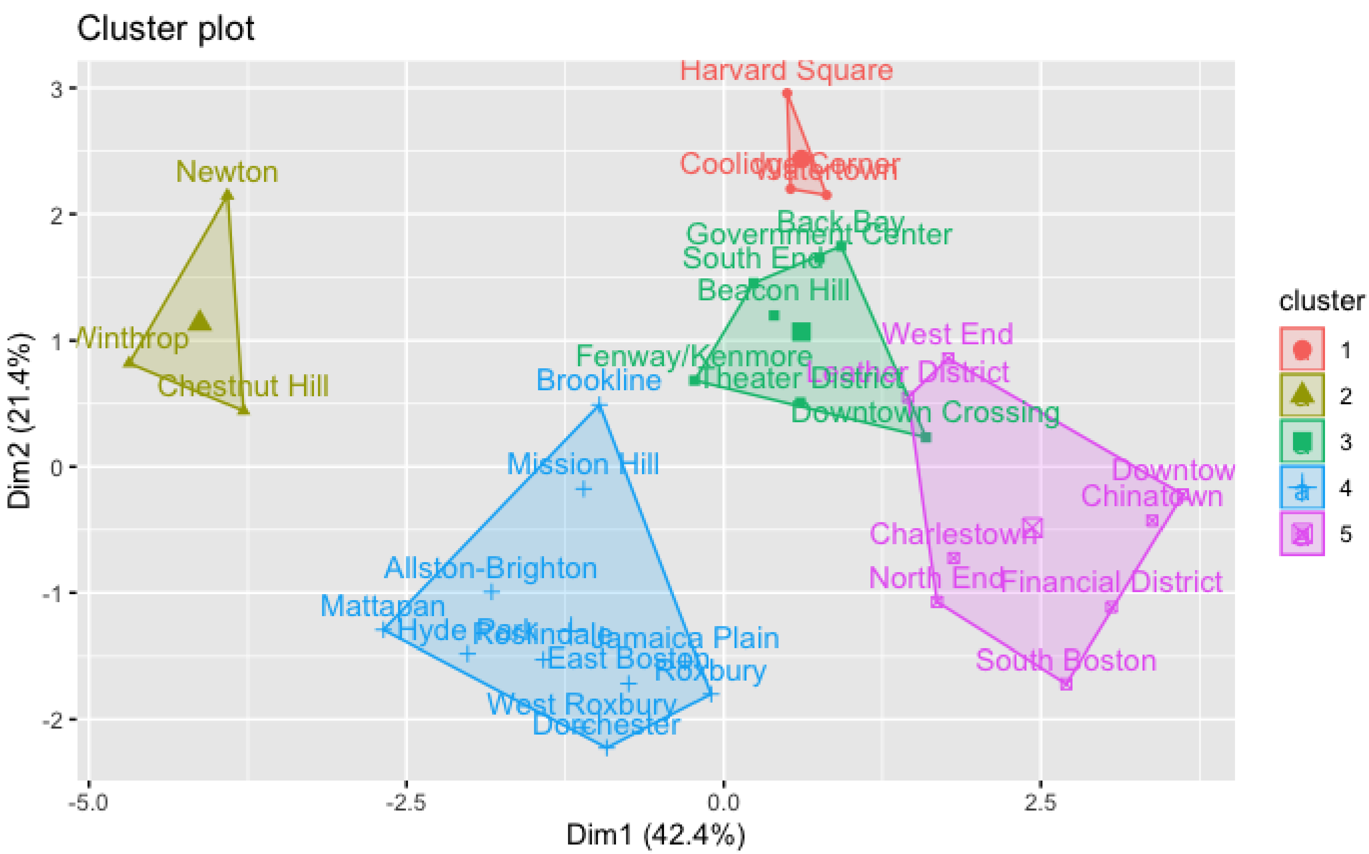
Hide

```
print(k4)
```

```
K-means clustering with 4 clusters of sizes 3, 10, 8, 11

Cluster means:
   log_price accommodates  bathrooms cancellation_policy number_of_reviews review_scores_rating   bedrooms
1 -1.2777191   -1.8855414 -1.1909272          -1.7572116        -1.2601805            1.0808591 -1.1637506
2  0.7411959    0.1455015 -0.6815972           0.4960988        -0.1633199           -0.3921439 -0.9893809
3  0.8740255    1.0619844  1.1303778           0.2483884         0.2006509            0.3117588  1.0584553
4 -0.9610005   -0.3903878  0.1223393          -0.1524055         0.3462304           -0.1650190  0.4470380
        beds nightly_price price_per_person
1 -1.78940736    -1.3474056       -0.8553313
2 -0.07369644     0.5835037        0.6883243
3  0.84539474     1.0538433        0.7997643
4 -0.05981559    -0.9294152       -0.9741239

Clustering vector:
  Allston-Brighton           Back Bay        Beacon Hill          Brookline        Charlestown 
                 4                  2                  2                  4                  3 
     Chestnut Hill          Chinatown    Coolidge Corner         Dorchester           Downtown 
                 1                  3                  2                  4                  3 
 Downtown Crossing        East Boston     Fenway/Kenmore Financial District  Government Center 
                 2                  4                  2                  3                  2 
    Harvard Square          Hyde Park      Jamaica Plain   Leather District           Mattapan 
                 2                  4                  4                  3                  4 
      Mission Hill             Newton          North End         Roslindale            Roxbury 
                 4                  1                  3                  4                  4 
      South Boston          South End   Theater District          Watertown           West End 
                 3                  2                  2                  2                  3 
      West Roxbury           Winthrop 
                 4                  1 

Within cluster sum of squares by cluster:
[1]  4.919815 52.736630 31.352756 38.045305
 (between_SS / total_SS =  59.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"        
[8] "iter"         "ifault"      
```

Hide

```
k5 <- kmeans(Scaled_Data, 5, nstart = 25)
p2 <- fviz_cluster(k5, data = Scaled_Data)
p2
```



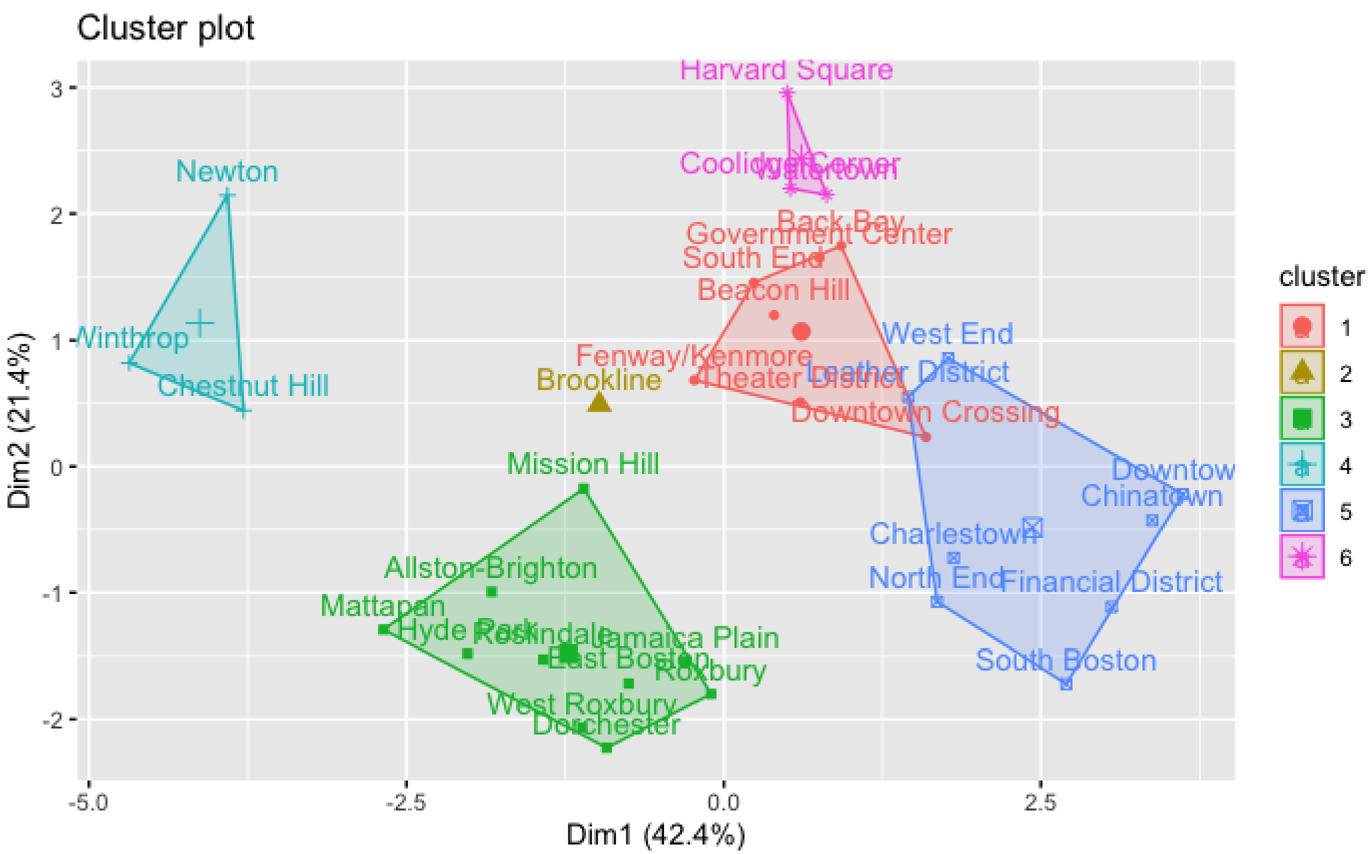
Hide

```
print(k5)
```

```
K-means clustering with 5 clusters of sizes 3, 3, 7, 11, 8

Cluster means:
   log_price accommodates  bathrooms cancellation_policy number_of_reviews review_scores_rating   bedrooms
1  1.0628585    0.5436913 -1.3573021          0.09577922        -1.3957647            0.9853532 -1.3922558
2 -1.2777191   -1.8855414 -1.1909272         -1.75721157        -1.2601805            1.0808591 -1.1637506
3  0.6033405   -0.0251513 -0.3920095          0.66766432         0.3648707           -0.9824998 -0.8167202
4 -0.9610005   -0.3903878  0.1223393         -0.15240548         0.3462304           -0.1650190  0.4470380
5  0.8740255    1.0619844  1.1303778          0.24838838         0.2006509            0.3117588  1.0584553
         beds nightly_price price_per_person
1  0.90633476     0.7781191        0.4305877
2 -1.78940736    -1.3474056       -0.8553313
3 -0.49370981     0.5000971        0.7987828
4 -0.05981559    -0.9294152       -0.9741239
5  0.84539474     1.0538433        0.7997643

Clustering vector:
  Allston-Brighton             Back Bay         Beacon Hill           Brookline         Charlestown
                 4                    3                   3                   4                   5
     Chestnut Hill            Chinatown     Coolidge Corner          Dorchester            Downtown
                 2                    5                   1                   4                   5
 Downtown Crossing          East Boston     Fenway/Kenmore Financial District  Government Center
                 3                    4                   3                   5                   3
    Harvard Square            Hyde Park       Jamaica Plain    Leather District            Mattapan
                 1                    4                   4                   5                   4
      Mission Hill               Newton           North End          Roslindale             Roxbury
                 4                    2                   5                   4                   4
      South Boston            South End    Theater District           Watertown            West End
                 5                    3                   3                   1                   5
      West Roxbury             Winthrop
                 4                    2

Within cluster sum of squares by cluster:
[1]  8.960427  4.919815 20.109007 38.045305 31.352756
 (between_SS / total_SS =  66.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

Hide

```
k6 <- kmeans(Scaled_Data, 6, nstart = 25)
p3 <- fviz_cluster(k6, data = Scaled_Data)
p3
```



Hide

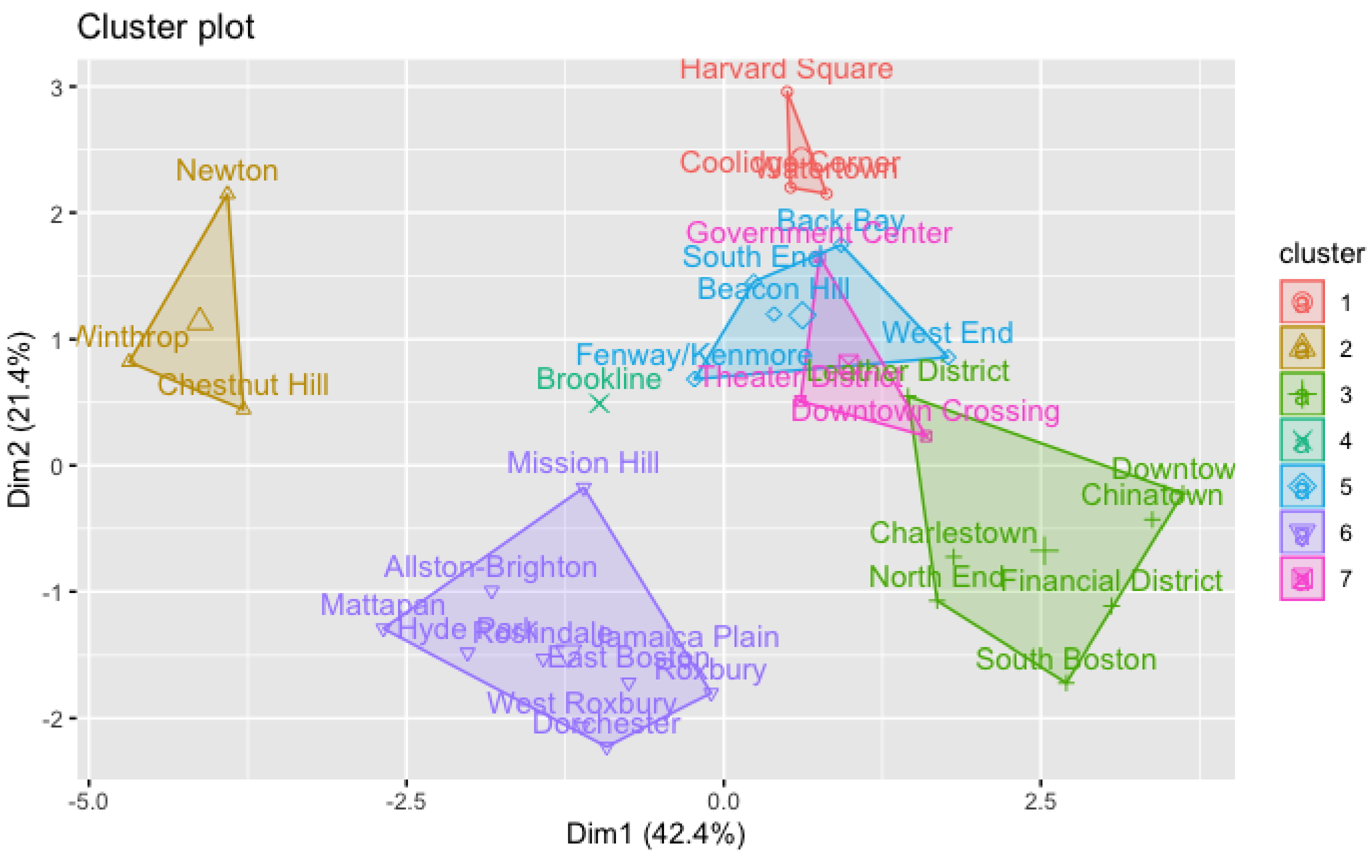
```
print(k6)
```

```
K-means clustering with 6 clusters of sizes 7, 1, 10, 3, 8, 3

Cluster means:
   log_price accommodates  bathrooms cancellation_policy number_of_reviews review_scores_rating   bedrooms
1  0.6033405   -0.0251513 -0.3920095          0.66766432         0.3648707           -0.9824998 -0.8167202
2 -0.2878330    0.2664419 -1.3573021          1.86454316        -1.3957647            0.9853532  1.0070489
3 -1.0283173   -0.4560708  0.2703034         -0.35410034         0.5204299           -0.2800562  0.3910369
4 -1.2777191   -1.8855414 -1.1909272         -1.75721157        -1.2601805            1.0808591 -1.1637506
5  0.8740255    1.0619844  1.1303778          0.24838838         0.2006509            0.3117588  1.0584553
6  1.0628585    0.5436913 -1.3573021          0.09577922        -1.3957647            0.9853532 -1.3922558
         beds nightly_price price_per_person
1 -0.49370981     0.5000971        0.7987828
2 -0.90038602    -0.7399561       -1.2765084
3  0.02424146    -0.9483611       -0.9438855
4 -1.78940736    -1.3474056       -0.8553313
5  0.84539474     1.0538433        0.7997643
6  0.90633476     0.7781191        0.4305877

Clustering vector:
  Allston-Brighton           Back Bay        Beacon Hill          Brookline        Charlestown
                 3                  1                  1                  2                  5
     Chestnut Hill          Chinatown    Coolidge Corner         Dorchester           Downtown
                 4                  5                  6                  3                  5
 Downtown Crossing        East Boston     Fenway/Kenmore Financial District  Government Center
                 1                  3                  1                  5                  1
    Harvard Square          Hyde Park      Jamaica Plain   Leather District           Mattapan
                 6                  3                  3                  5                  3
      Mission Hill             Newton          North End         Roslindale            Roxbury
                 3                  4                  5                  3                  3
      South Boston          South End   Theater District          Watertown           West End
                 5                  1                  1                  6                  5
      West Roxbury           Winthrop
                 3                  4

Within cluster sum of squares by cluster:
[1] 20.109007  0.000000 24.133160  4.919815 31.352756  8.960427
 (between_SS / total_SS =  71.1 %)

Available components:
```

```
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

Hide

```
k7 <- kmeans(Scaled_Data, 7, nstart = 25)
p4 <- fviz_cluster(k7, data = Scaled_Data)
p4
```



Hide

```
print(k7)
```

```
K-means clustering with 7 clusters of sizes 3, 3, 7, 1, 5, 10, 3

Cluster means:
   log_price accommodates   bathrooms cancellation_policy number_of_reviews review_scores_rating   bedrooms
1  1.0628585    0.5436913 -1.35730206          0.09577922       -1.39576469            0.9853532 -1.3922558
2 -1.2777191   -1.8855414 -1.19092717         -1.75721157       -1.26018046            1.0808591 -1.1637506
3  0.8589479    1.1774613  1.10181853          0.18661073        0.39085095            0.4153587  1.1659898
4 -0.2878330    0.2664419 -1.35730206          1.86454316       -1.39576469            0.9853532  1.0070489
5  0.6112883   -0.2449134  0.06828808          0.37879885       -0.02660866           -0.3054778 -0.5410740
6 -1.0283173   -0.4560708  0.27030339         -0.35410034        0.52042986           -0.2800562  0.3910369
7  0.7155037    0.4340513 -0.58507142          1.15349597        0.51879607           -1.9211835 -0.9019859
        beds nightly_price price_per_person
1  0.90633476     0.7781191        0.4305877
2 -1.78940736    -1.3474056       -0.8553313
3  1.02728813     1.0555134        0.7320034
4 -0.90038602    -0.7399561       -1.2765084
5 -0.67760477     0.5984635        1.1614178
6  0.02424146    -0.9483611       -0.9438855
7 -0.16526795     0.5168383        0.3528272

Clustering vector:
  Allston-Brighton           Back Bay        Beacon Hill          Brookline        Charlestown
                 6                  5                  5                  4                  3
     Chestnut Hill          Chinatown    Coolidge Corner         Dorchester           Downtown
                 2                  3                  1                  6                  3
 Downtown Crossing        East Boston     Fenway/Kenmore Financial District  Government Center
                 7                  6                  5                  3                  7
    Harvard Square          Hyde Park      Jamaica Plain   Leather District           Mattapan
                 1                  6                  6                  3                  6
      Mission Hill             Newton          North End         Roslindale            Roxbury
                 6                  2                  3                  6                  6
      South Boston          South End   Theater District          Watertown           West End
                 3                  5                  7                  1                  5
      West Roxbury           Winthrop
                 6                  2

Within cluster sum of squares by cluster:
[1]  8.960427  4.919815 24.949347  0.000000  9.446952 24.133160  7.013507
 (between_SS / total_SS =  74.4 %)
```

```
Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

Hide

```
# comparing the clusters
# grid.arrange(p1, p2, p3, p4, nrow = 2)

# Adding to our initial data to do some descriptive statistics at the cluster level
k4$centers
```

```
   log_price accommodates  bathrooms cancellation_policy number_of_reviews review_scores_rating   bedrooms
1 -1.2777191   -1.8855414 -1.1909272          -1.7572116        -1.2601805            1.0808591 -1.1637506
2  0.7411959    0.1455015 -0.6815972           0.4960988        -0.1633199           -0.3921439 -0.9893809
3  0.8740255    1.0619844  1.1303778           0.2483884         0.2006509            0.3117588  1.0584553
4 -0.9610005   -0.3903878  0.1223393          -0.1524055         0.3462304           -0.1650190  0.4470380
        beds nightly_price price_per_person
1 -1.78940736    -1.3474056       -0.8553313
2 -0.07369644     0.5835037        0.6883243
3  0.84539474     1.0538433        0.7997643
4 -0.05981559    -0.9294152       -0.9741239
```

Hide

```
Boston %>%
  mutate(Cluster = k4$cluster) %>%
  group_by(Cluster) %>%
  summarise_all("mean")
```

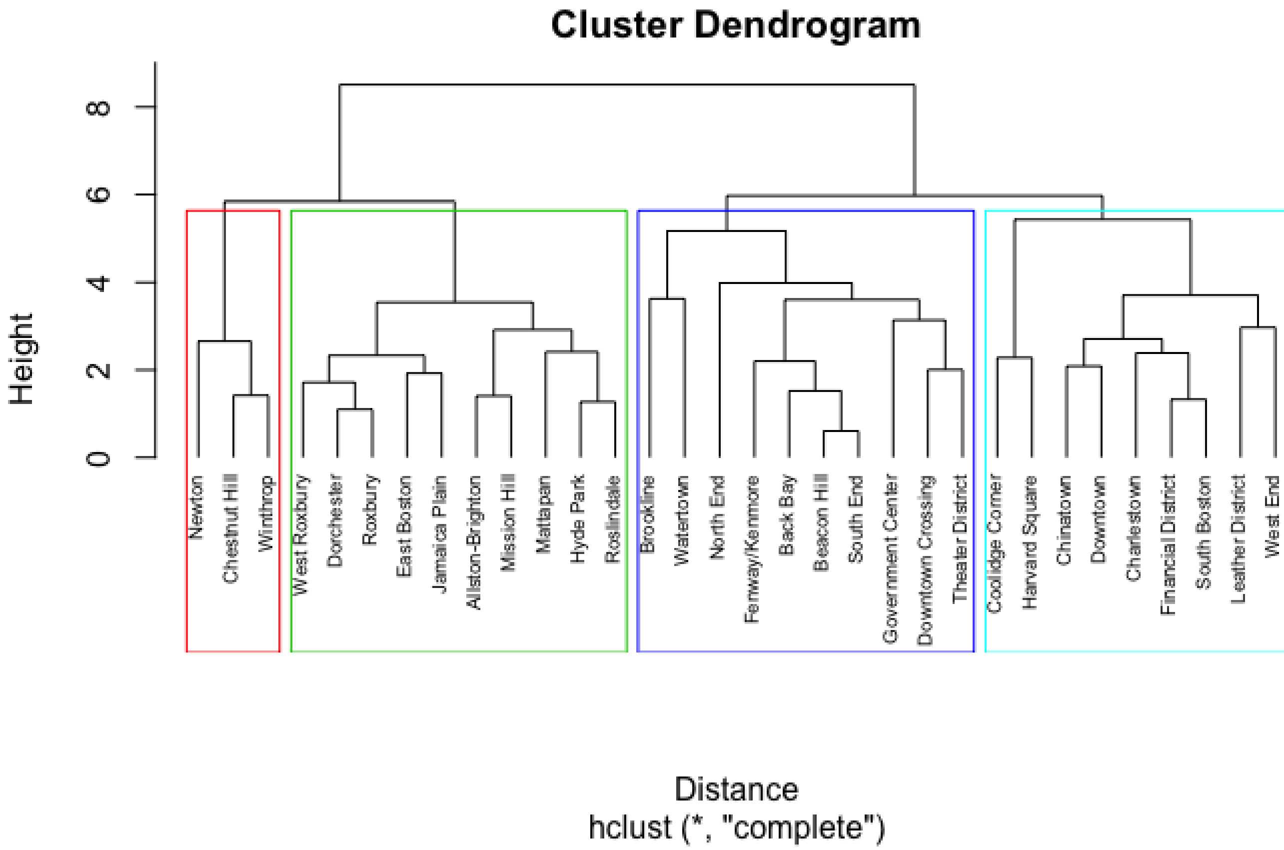
Cluster <int>	log_price <dbl>	accommodates <dbl>	bathrooms <dbl>	cancellation_policy <dbl>	number_of_reviews <dbl>
1	4.376402	2.206349	1.023810	1.634921	1.904762
2	5.256466	3.427297	1.096699	2.484218	17.314063
3	5.314367	3.978235	1.356006	2.390853	22.427325

Cluster	log_price	accommodates	bathrooms	cancellation_policy	number_of_reviews
<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
4	4.514463	3.105151	1.211748	2.239790	24.472506

4 rows | 1-6 of 11 columns

Hide

```
# Hierarchical Clustering
# Dissimilarity matrix
Distance <- dist(Scaled_Data, method = "euclidean")
# Hierarchical clustering using Complete Linkage
hc <- hclust(Distance, method = "complete" )
# Plot the obtained dendrogram
plot(hc, cex = 0.6, hang = -1)
rect.hclust(hc, k = 4, border = 2:5)
```



Step V: Conclusions

Code

```
[1] "The overall process was very collaborative in nature. As a team we had decided to prepare and explore the da
ta together and then work on our individual areas to come up with our analysis. In the process of doing so we had
discussed and commented constructively on each other's work which resulted in a much better quality output in the
end. The exploratory analysis of the AirBnb data helped us understand the rental landscape of Boston through vari
ous statistics and visualizations. For e.g. the clustering analysis shows how certain neighborhoods are similar i
n nature and also what are the various characteristics that make them a part of each cluster. This helps individu
als and businesses alike to answer certain questions like which neighborhoods have the highest review scores, lis
ting price etc. \n\nThe classification tree model could be useful for a property owner who is interested in listi
ng their property for rental on AirBnb. If they were not sure what type of cancelation policy they should impleme
nt for their new rental, AirBnb could provide a service that helped them with “Based on the characteristics that
you have provided regarding your potential listing, and other properties that share some of these characteristic
s, we recommend a cancelation policy of “x”.” If you operate under the presumption that the existing properties f
rom which the model was built have their cancellation policies for good reason, this will help the new owner arri
ve at a good decision from the get-go. The property owner, AirBnb, and even potentially the ultimate customer/ren
ter can all benefit from implementation of a Classification Tree such as the one we constructed. \n\nWhen we exam
ine the Naive Bayes model, we know that the model with its selected variables is pretty accurate. Upon setting up
the categorical bins derived from the log price variable, we can see that the majority of AirBnb rentals were con
ducted in the below average price category. This plays an important role in realizing how much visitors are willi
ng to pay to stay in Boston. With the mean and median log price falling at 4.913/4.884 we can see that the majori
ty would rather pay less than the median/mean log price. In its true dollar format, this comes out to be around
$80 a night but less than $136 dollars. \n\nAn important question to consider is who are these visitors? By using
Naive Bayes alone it is difficult to determine who these visitors are. However, when you combine our Naive Bayes
model along with other models such as the K-Means Analysis, you will be able to see which neighborhoods the ‘belo
w average price category’ would be placed in and depict a clearer picture of who the audience are and their need
s. Boston is known for being a college town with most colleges centered at or around the city. If there is anyone
who is willing to pay a below average price rating, it would most likely be college students which can be seen as
the majority of would fall at or below average price category. But you also have another half that is unknown and
that half for certain falls in above average category.\n\nWe can therefore conclude that the Airbnb data does giv
e an insight as to who is visiting Boston and what is the price that they are willing to pay to stay. By using va
riables such as the price, cancellation policy, cleaning fee, neighborhoods, etc., we can derive further in depth
questions as to the demographics of renting in Boston.\n"
```