Airbnb in Southbank, Melbourne

Presented by

Garfield Consulting Service

Guorong Liu, Fangwen Wang

Duosi Zheng, Yuanzhe Zhuang

Metropolitan College | Boston University | Dec 10, 2019

Agenda

1. **Data Preparation & Exploration**
2. **Prediction: Multiple Regression Model**
3. **Classification**
   - A. **K-Nearest Neighbors**
   - B. **Naïve Bayes**
   - C. **Classification Tree**
4. **Clustering**
5. **Conclusion & Suggestions**

1. **Data Preparation & Exploration:** Missing Values

Methodologies of Data Cleaning:

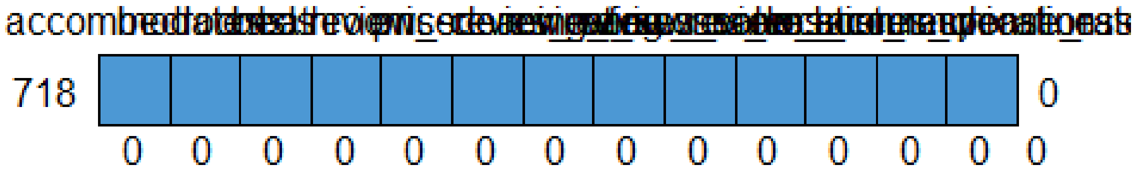
I. <u>dropping</u> NA values, N/A values, and blank cells for most of the data models

II. <u>mean values</u> : substitute the values of missing values (i.e. clustering model)

III. Use sum(is.na(df) and md.pattern() to check null values

IV. Dummy variables ← categorical values

```
# filter the only neighborhood Southbank
df1 <- filter(df, neighborhood=="Southbank")
dim(df1)
## [1] 1248   84
1248*84
## [1] 104832
#tell us the missing values,
this gives the number of null values:
sum(is.na(df1))
## [1] 5156
```



(example for checking a random data frame)

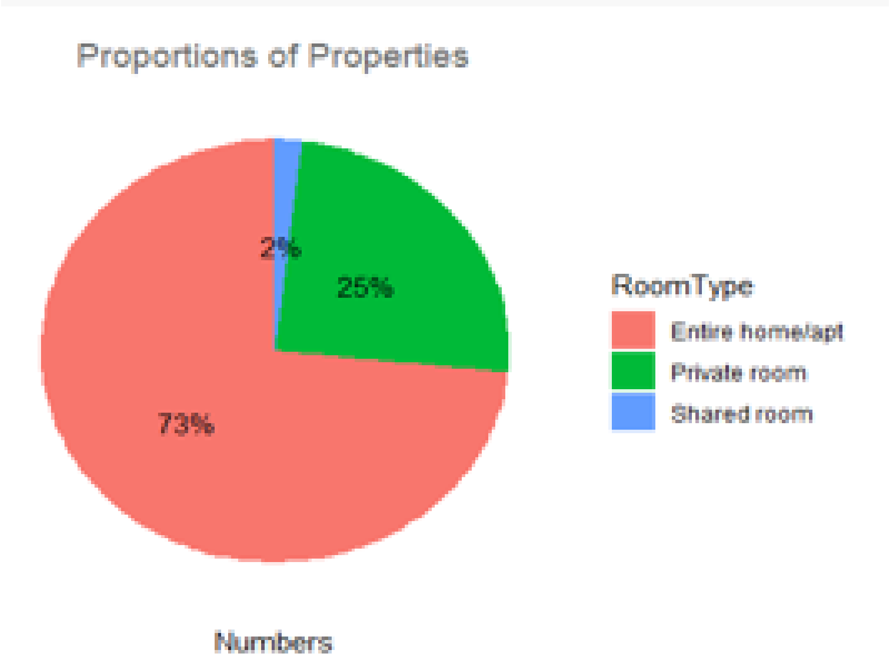
1. Data Preparation & Exploration: Summary Statistic & Visualizations

Who can tell any words?

Word Cloud

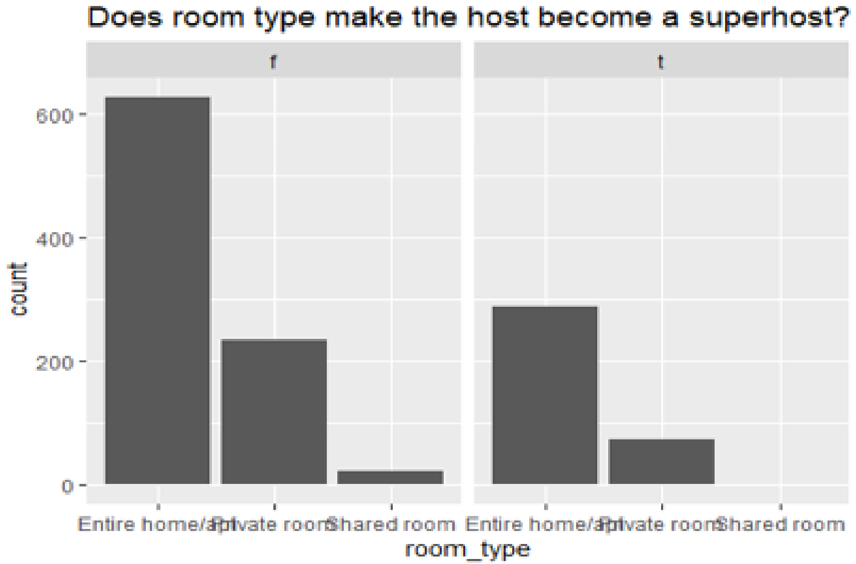
Summaries of KPIs
[i.e. Review Scores Value (total:10)]

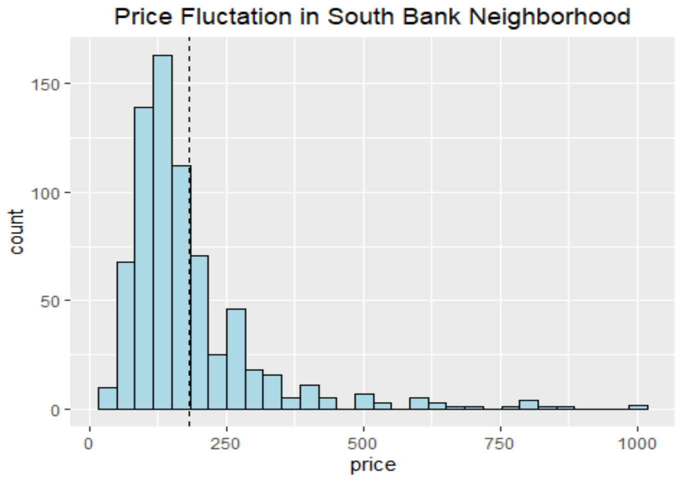
Mean	**9.48**
Standard Deviation	**0.85**
Median	**10**
Maximum	**10**
Minimum	**2**

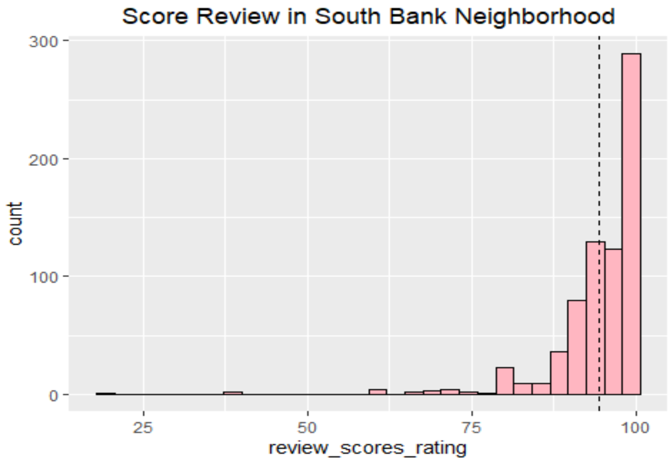


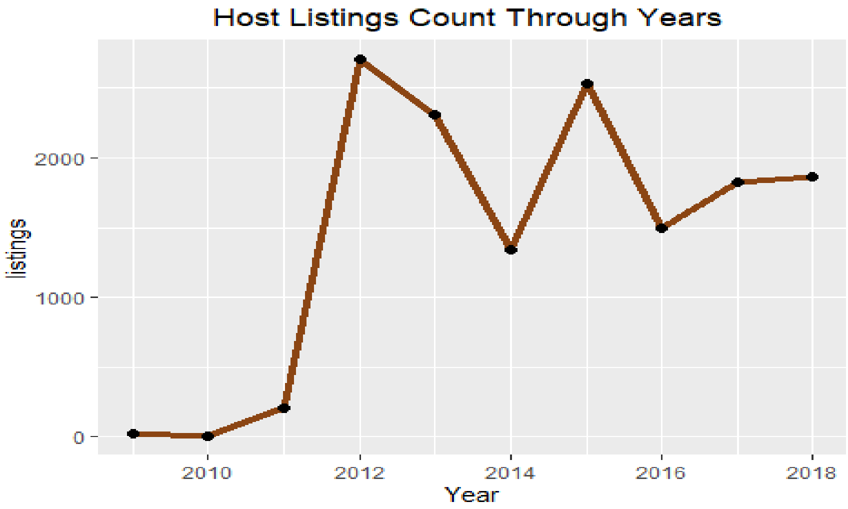
Most of the rooms are **entire home/apt**.

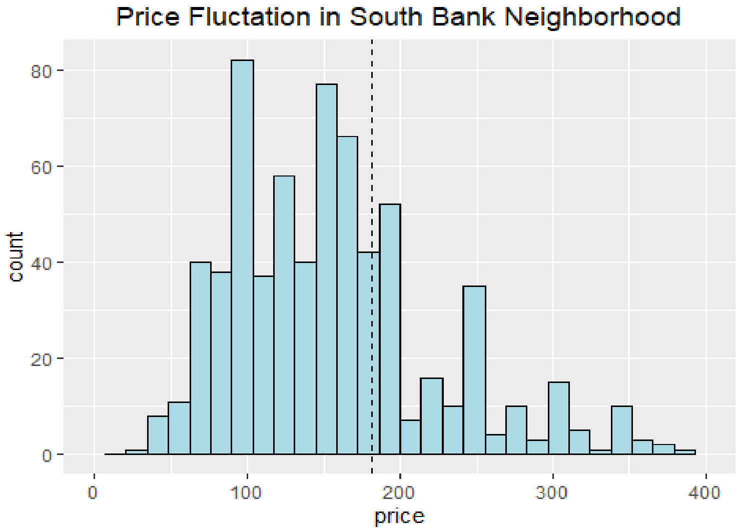
1. Data Preparation & Exploration: Summary Statistic & Visualizations

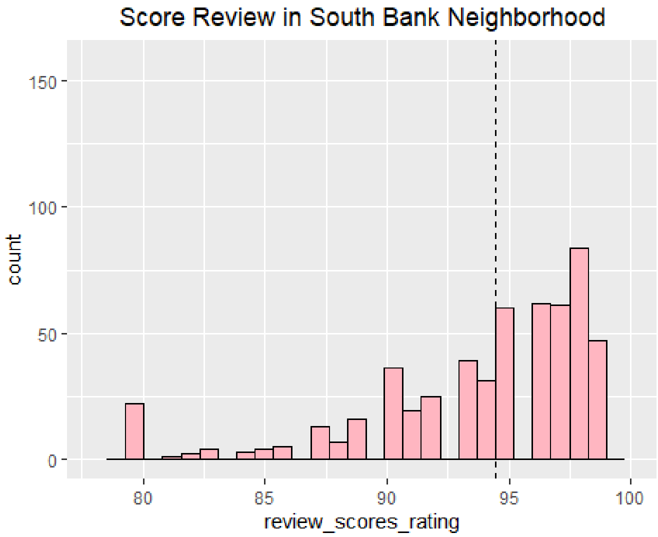












2. Prediction: Multiple Regression Model

```
> summary(model1)

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

Residuals:
    Min      1Q  Median      3Q     Max 
-185.85  -46.64  -13.40   26.39  910.49 

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   -29.09764   54.67879  -0.532 0.594880    
accommodates                   21.85964    3.91621   5.582 4.11e-08 ***
bathrooms                      57.32668   11.19271   5.122 4.49e-07 ***
cleaning_fee                    0.16204    0.12196   1.329 0.184619    
host_is_superhost              -7.61686   10.32731  -0.738 0.461174    
guests_included                 9.70122    3.72047   2.608 0.009422 ** 
number_of_reviews               0.05668    0.11195   0.506 0.612888    
review_scores_value             1.37148    5.60424   0.245 0.806783    
reviews_per_month             -13.85994    3.14882  -4.402 1.34e-05 ***
security_deposit                0.02743    0.01488   1.843 0.065946 .  
instant_bookable                5.49858    9.86502   0.557 0.577543    
availability_30                 1.95821    0.53451   3.664 0.000278 ***
calculated_host_listings_count -0.79404    0.25108  -3.162 0.001670 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 98.19 on 452 degrees of freedom
Multiple R-squared:  0.4363,	Adjusted R-squared:  0.4213 
F-statistic: 29.15 on 12 and 452 DF,  p-value: < 2.2e-16
```

```
> summary(model2)

Call:
lm(formula = price ~ accommodates + bathrooms + guests_included + 
    reviews_per_month + security_deposit + availability_30 + 
    calculated_host_listings_count, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-190.93  -47.03  -14.38   27.54  900.42 

Coefficients:
                               Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   -14.17781   15.54291  -0.912 0.362159    
accommodates                   24.10600    3.58895   6.717 5.54e-11 ***
bathrooms                      57.58753   11.04951   5.212 2.84e-07 ***
guests_included                 9.35797    3.64854   2.565 0.010640 *  
reviews_per_month             -13.03112    2.44978  -5.319 1.63e-07 ***
security_deposit                0.02877    0.01448   1.987 0.047495 *  
availability_30                 1.94863    0.53128   3.668 0.000273 ***
calculated_host_listings_count -0.71881    0.23872  -3.011 0.002748 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 97.93 on 457 degrees of freedom
Multiple R-squared:  0.4331,	Adjusted R-squared:  0.4244 
F-statistic: 49.88 on 7 and 457 DF,  p-value: < 2.2e-16
```

Cleaning Data → All variables in model 1 → Eliminate multicollinearity by viewing the correlation table → Backward elimination → Selective variables in Model 2

2. Prediction: Multiple Regression Model

i. Generate Equation

ii. Predict Values

iii. Values Match: 138 AUD

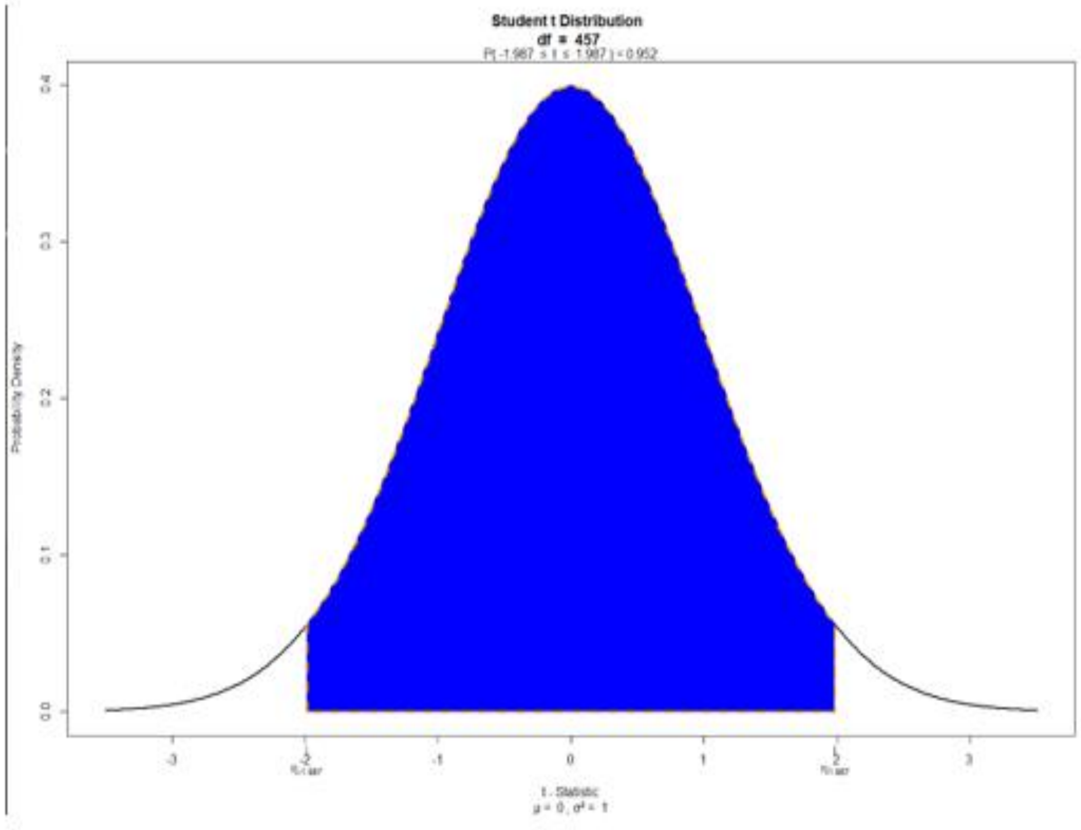
```
> predict(model2, myhouse)
       1
138.1482
> -14.177 + 24.11*2 + 57.59*2 + 9.36*2 + -13.03*5 + 0.029*0 + 1.95*20 + -0.72*5
[1] 138.193
>
```

visualize.t(stat=c(-1.987, 1.987), df=457,

section = "bounded")

→ **<u>0.952</u>** of the shaded area →

very reliable



2. Prediction: Multiple Regression Model

R-Squared: 0.4363 → 0.4331

- sum of squared error divides by the sum of square total.
- Adjusted R-Squared: 0.4331 → 0.4244
- Less redundancy
- More efficient variations

RMSE
(the square root of mean squared error)

```
> accuracy(pred1, train$price)
                   ME    RMSE      MAE       MPE     MAPE
Test set 1.404686e-12 97.0806 56.93222 -15.00545 35.50696
> pred2 <- predict(model2, valid)
> accuracy(pred2, valid$price)
               ME     RMSE      MAE       MPE     MAPE
Test set 5.709778 97.81584 57.33693 -10.39859 33.33211
> 
```

- Less error in train data: 97.0806 > 97.81584
- Not much differences in two RMSE

We have a RELIABLE model!

3. **Classification:** K-nearest Neighbors

<u>Outcome</u>: What kind of cancellation policy?

<u>Variables</u>:

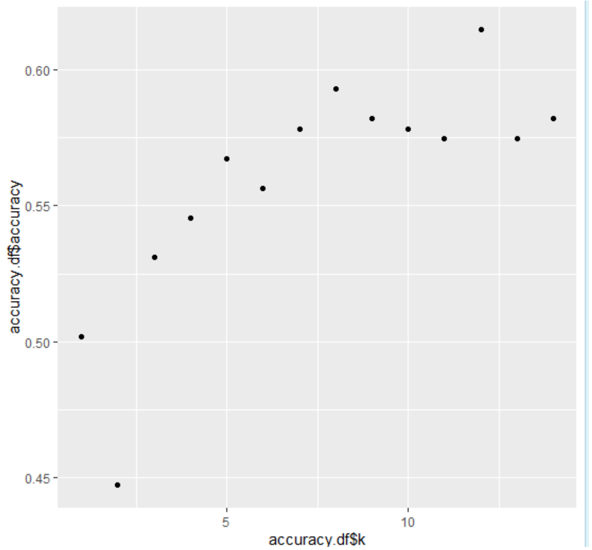
i. Host response rate

ii. Price

iii. Security Deposit

iv. Review scores communication

```
##      k  accuracy
## 1    1 0.5018182
## 2    2 0.4472727
## 3    3 0.5309091
## 4    4 0.5454545
## 5    5 0.5672727
## 6    6 0.5563636
## 7    7 0.5781818
## 8    8 0.5927273
## 9    9 0.5818182
## 10  10 0.5781818
## 11  11 0.5745455
## 12  12 0.6145455
## 13  13 0.5745455
## 14  14 0.5818182
```



K-value: 12 | Accuracy: 61.45%

Flexible Cancellation Policy

```
> # build knn model
> nn <- knn(train = train.norm.df[, 2:5], test = new.rental.norm.df,
  = 7)
> row.names(train.df)[attr(nn, "nn.index")]
[1] "109" "16"  "394" "211" "273" "174" "133"
>
```

flexible (109), flexible (16), flexible (394), flexible (211), flexible (273), moderate (174), flexible (133)

FLEXIBLE

```
> row.names(train.df)[attr(knn12,"nn.index")]
 [1] "109" "16"  "394" "211" "273" "174" "133" "161" "370" "223" "282" "106"
>
> new.rental.neighbor_valid <- train.df[c(109,16,394,211,174,133,161,370,223,282,106
> new.rental.neighbor_valid$cancellation_policy #my policy is flexible
 [1] flexible                    flexible                    flexible
 [4] flexible                    moderate                    flexible
 [7] flexible                    strict_14_with_grace_period moderate
[10] flexible                    moderate
```

3. Classification: Naïve Bayes

<u>Outcome</u>: What can measure to become instant bookable housing?

<u>Variables</u>:

i. Host response rate
ii. Number of beds
iii. Available days in a month
iv. Cancellation Policy

```
A-priori probabilities:
Y
        f         t
0.3788927 0.6211073

Conditional probabilities:
   cancellation_policy
Y      flexible    moderate      strict strict_14_with_grace_period super_strict_30
  f 0.237442922 0.333333333 0.000000000                 0.406392694     0.000000000
  t 0.105849582 0.242339833 0.000000000                 0.649025070     0.000000000
   cancellation_policy
Y   super_strict_60
  f     0.022831050
  t     0.002785515

   beds
Y       [,1]     [,2]
  f 2.118721 1.460353
  t 2.406685 1.517702

   availability_30
Y       [,1]     [,2]
  f 10.70776 9.918312
  t 12.58217 8.624268

   host_response_rate
Y          [,1]         [,2]
  f 0.009538813 0.0013178383
  t 0.009725905 0.0007377852
```

```
ficational<-data.frame(beds=3,
                       cancellatic _p li cy= s.factor("flexible"),
                       availabilit y_30= 4,
                       host_response_r te=1)
predict(naive,ficational)

[1] t
Levels: f t
```

True for instant bookable!

```
               Accuracy : 0.6419
                 95% CI : (0.6013, 0.681)
    No Information Rate : 0.6211
    P-Value [Acc > NIR] : 0.1621

                  Kappa : 0.1062
```

Training

```
               Accuracy : 0.6078
                 95% CI : (0.557, 0.6569)
    No Information Rate : 0.5922
    P-Value [Acc > NIR] : 0.285

                  Kappa : 0.0855
```

validation

<u>Results for the Conditions:</u>

[accuracy 64.19% in training set]

a) faster host response
b) enough bed for at least one or two people
c) more available housing,
d) somehow flexible

3. **Classification:** Classification Tree

Purpose: Predict the size of the cleaning fee.

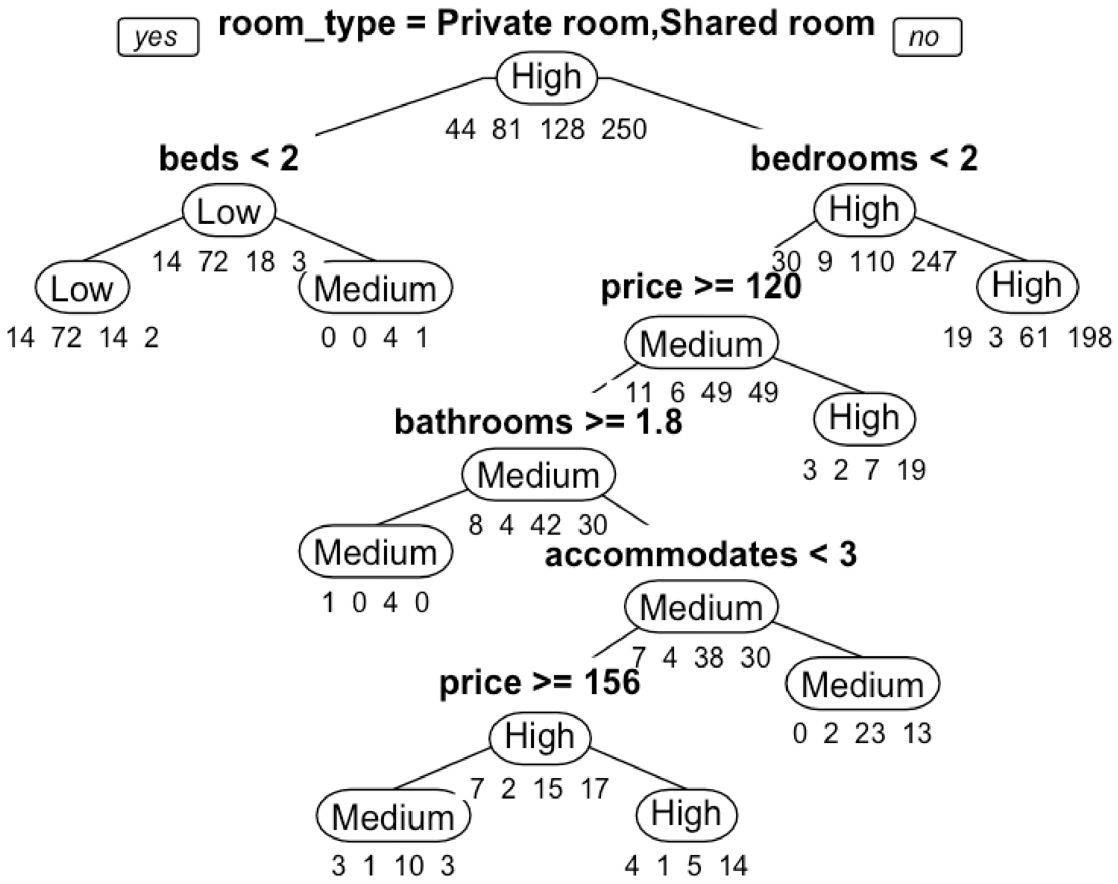
Variables:

- Room type
- Accommodates
- # of bathrooms, bedrooms and beds
- Price

Bin the Cleaning Fees

- **No fee**
- **Low 1-35**
- **Medium 36-73**
- **High 74-450**

3. Classification: Classification Tree

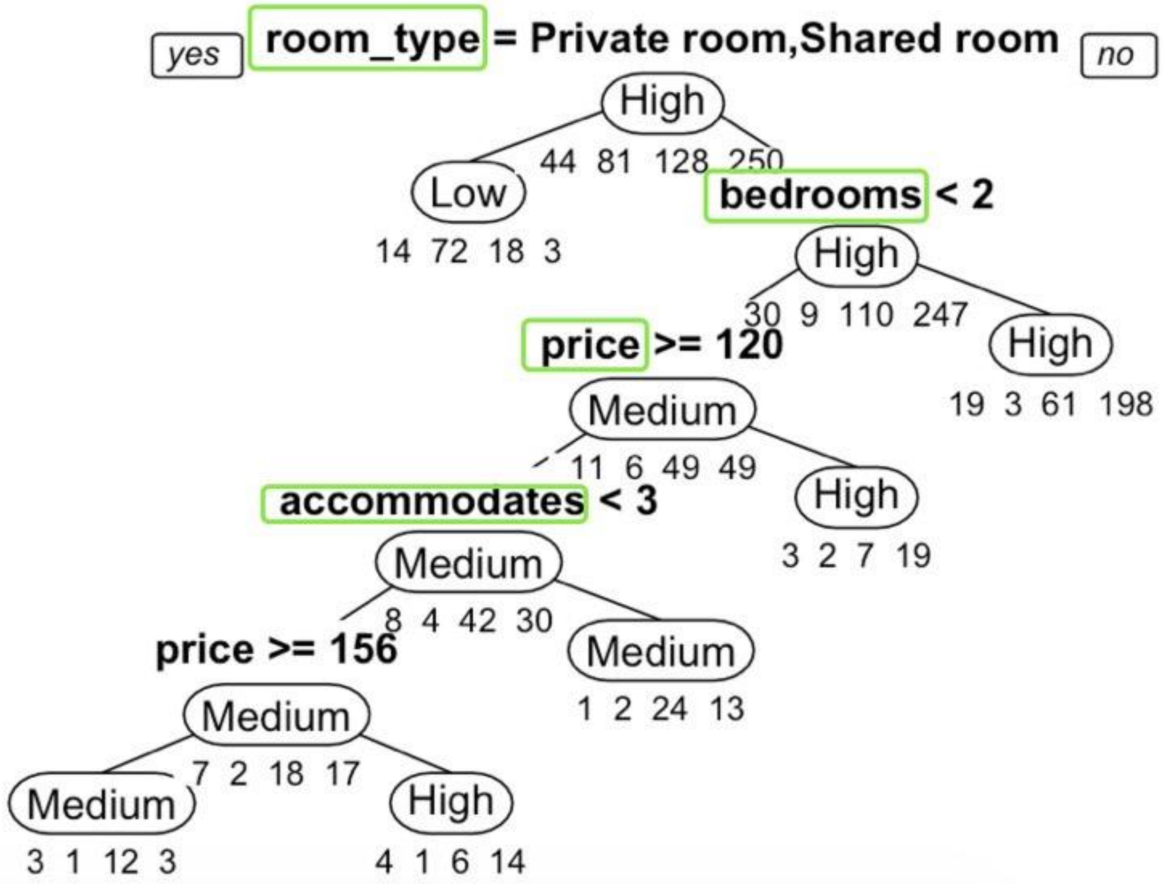


Cross-Validation

- Determine the ideal size
- Find the minimum xerror
- Get the optimal cp value is 0.0039526

3. Classification: Classification Tree

Classification Tree with ideal size



For instance, the bottom-left-most terminal node

= "Medium" cleaning fee →

if it is the entire room or apartment and the

bedrooms < 2 and,

the price is not smaller than 120 and,

accommodates < 3 and,

the price is not smaller than 156.

3. Classification: Classification Tree

Confusion matrix

Training set

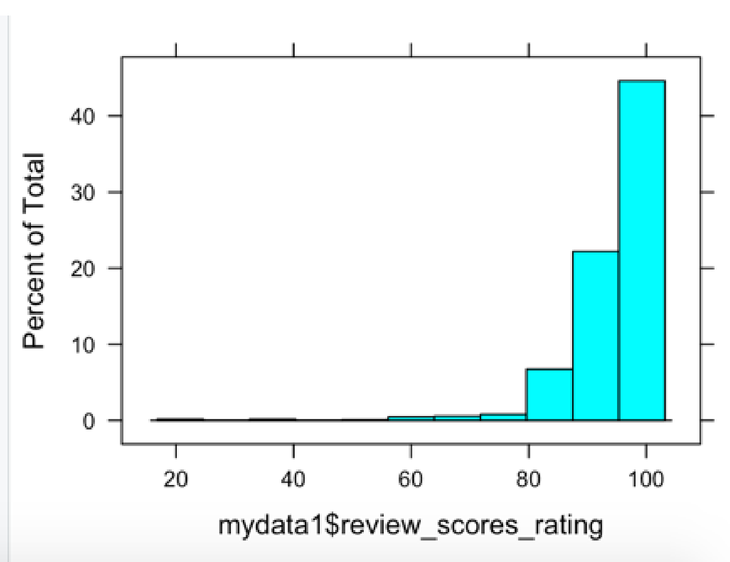
```
Confusion Matrix and Statistics

          Reference
Prediction No Fee Low Medium High
    No Fee      0   0      0    0
    Low        14  72     18    3
    Medium      4   3     36   16
    High       26   6     74  231

Overall Statistics

               Accuracy : 0.674
                 95% CI : (0.6311, 0.7148)
    No Information Rate : 0.497
    P-Value [Acc > NIR] : 7.913e-16

                  Kappa : 0.4592

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: No Fee Class: Low Class: Medium Class: High
Sensitivity                0.00000     0.8889       0.28125      0.9240
Specificity                1.00000     0.9171       0.93867      0.5810
Pos Pred Value                 NaN     0.6729       0.61017      0.6855
Neg Pred Value             0.91252     0.9773       0.79279      0.8855
Prevalence                 0.08748     0.1610       0.25447      0.4970
Detection Rate             0.00000     0.1431       0.07157      0.4592
Detection Prevalence       0.00000     0.2127       0.11730      0.6700
Balanced Accuracy          0.50000     0.9030       0.60996      0.7525
```

Validation set

```
Confusion Matrix and Statistics

          Reference
Prediction No Fee Low Medium High
    No Fee      0   0      0    0
    Low         8  40     14    2
    Medium      6   3     13   10
    High       15   8     61  154

Overall Statistics

               Accuracy : 0.6198
                 95% CI : (0.5653, 0.6721)
    No Information Rate : 0.497
    P-Value [Acc > NIR] : 4.270e-06

                  Kappa : 0.357

 Mcnemar's Test P-Value : 2.022e-14

Statistics by Class:

                     Class: No Fee Class: Low Class: Medium Class: High
Sensitivity                0.00000     0.7843       0.14773      0.9277
Specificity                1.00000     0.9152       0.92276      0.5000
Pos Pred Value                 NaN     0.6250       0.40625      0.6471
Neg Pred Value             0.91317     0.9593       0.75166      0.8750
Prevalence                 0.08683     0.1527       0.26347      0.4970
Detection Rate             0.00000     0.1198       0.03892      0.4611
Detection Prevalence       0.00000     0.1916       0.09581      0.7126
Balanced Accuracy          0.50000     0.8498       0.53525      0.7139
```

4. Clustering

- <u>Special Data Cleaning</u>: mean value → substitution → missing values



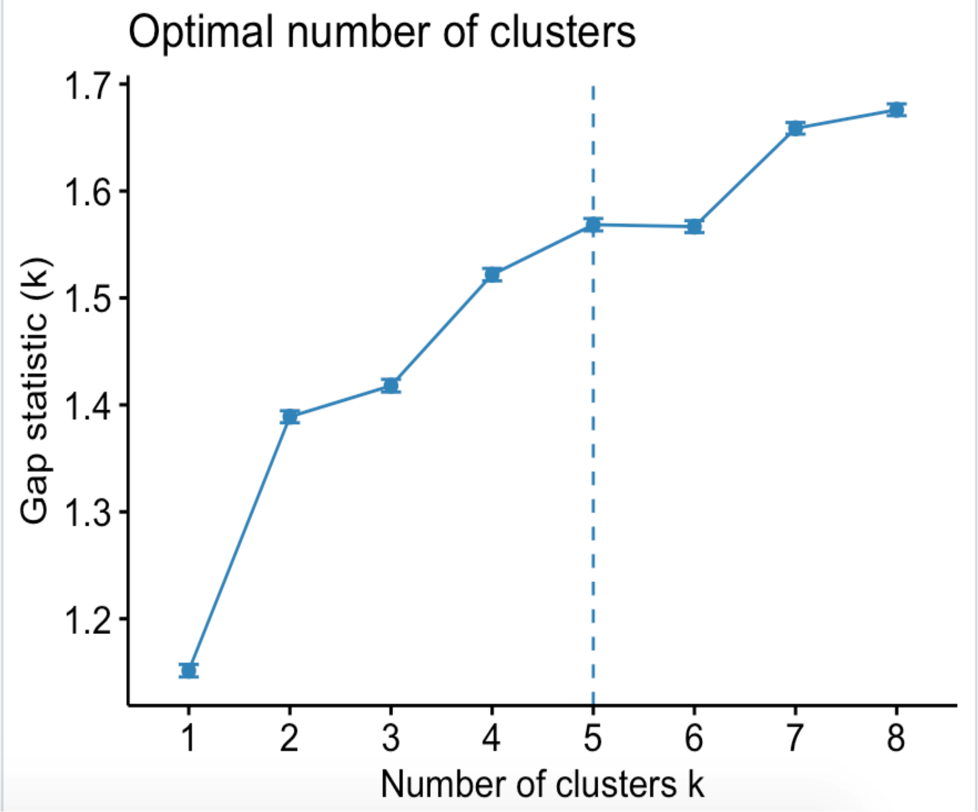

```
> histogram(mydata1$review_scores_rating)
> summary(mydata1$review_scores_rating)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  20.00   92.00   96.00   94.23   99.00  100.00     302
```

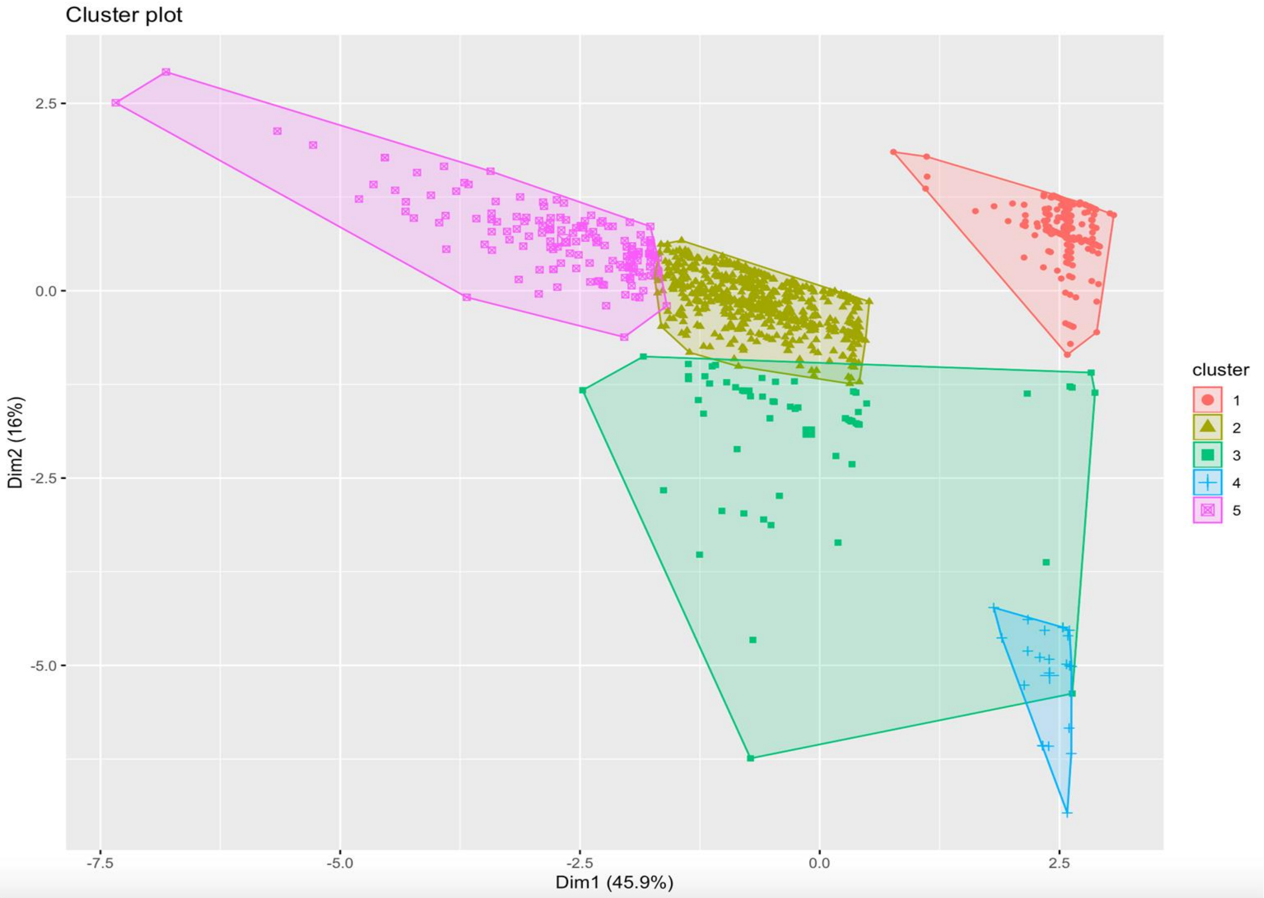
- <u>Selected Variables</u>: accommodates, bathrooms, bedrooms, beds, price, review score of rating, room type

- <u>Feature Engineering</u>:
  - Categorical Variables → Dummy Variables
  - Introduce "weight" to describe the related variables

```
> # a dataframe includes numerica variables
> accommodates <- mydata1$accommodates
> comfort <- mydata1$bathrooms*.35 + mydata1$bedrooms*.3 + mydata1$beds*.35
> price <- mydata1$price
> evaluation <- mydata1$review_scores_rating
> mydata2 <- data.frame(accommodates,comfort,price,evaluation)
```

4. Clustering





Optimal Cluster Groups: 5

4. Clustering

Categories	Attributes	Recommendations		
		Room Type	Surrounding	Promotions
Rich Couples	Enjoy high-level life	Private, Fancy, High reviewing score	Fine restaurant, Luxury places	Club member
Close Friends	Go out together	Entire room or Apartment, Big, High reviewing score	Nice plaice for taking pictures and hanging out	Discount for restaurants, Uber or Lift
Thrifty People	Save Money	Entire room or Apartment, Lower price	Free places	Awards from completing task
Colleagues	Go out for business	Shared	Convenient Transportation	Discount for room service
Family	Go out together	Entire room or Apartment, Big, High reviewing score	Nice plaice for taking pictures and hanging out	Discount for restaurants, renting car

Conclusion & Recommendations

airbnb

- Value Privacy
- Suitable for Vacations
- Advice for Host: Flexible Cancellation, Faster Host Response, Cheaper Housing
- Relaxing Neighborhood → Investment Opportunities
- More Diverse Property Type

airbnb

Any Questions?