# AD 699: Semester Project


**Team members:**

Zehuang Hong Chen
Vasinee Powthong
Junpeng Li
Yang Xiao
Amirali Abolhelm

# Agenda

- Data Preparation & Exploration
- Prediction
- Classification
- Clustering
- Conclusions
- Q&A

# Data Preparation & Exploration

- Data simplification
  - **FILTER** NYC as object
  - **DELETE** meaningless column
  - **SUBSET** conditional algorithm
- Data summary statistic functions
  - Summary()
  - Rd()
  - Range()
  - stat.desc()
  - Etc.

```
NYC = filter(airbnb, city == "NYC")
NYC = NYC[-c(20,21,26)]
NYC$property_type = droplevels(NYC$property_type)
```

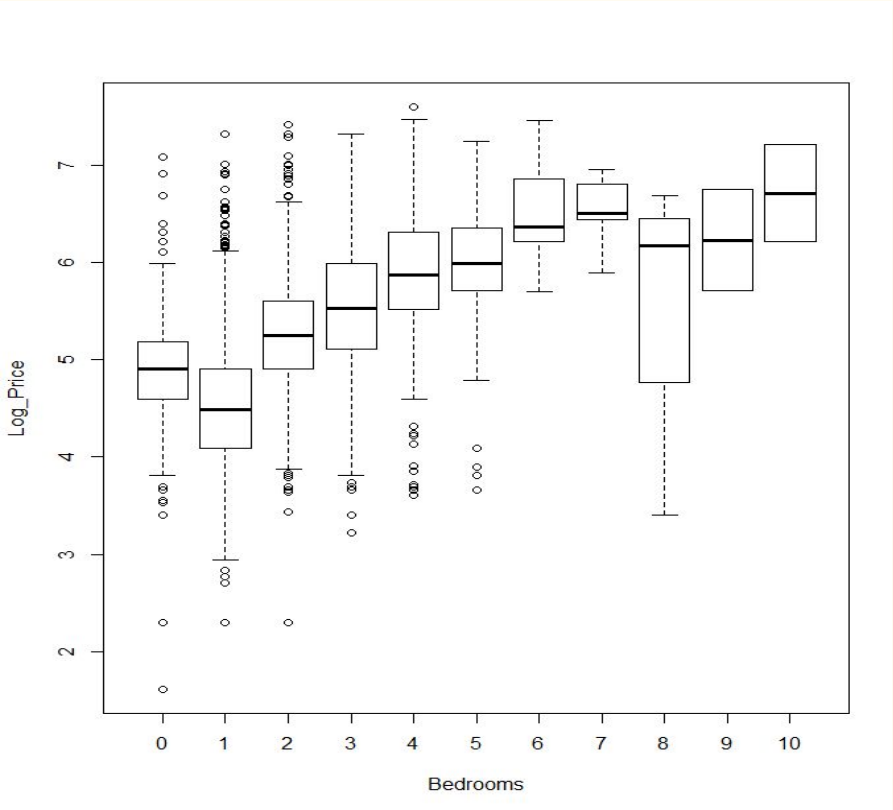
```
anyNA(NYC)
library(tidyr)
NYC[NYC == ""] = NA
NYC = drop_na(NYC)
min(NYC$log_price)
library(dplyr)
NYC = subset(NYC, log_price != "0")
min(NYC$log_price)
```

```
summary(NYC$log_price)
sd(NYC$log_price)
range(NYC$log_price)
summary(NYC$neighbourhood)
library(pastecs)
stat.desc(NYC$log_price)
stat.desc(airbnb)
```

```
> stat.desc(NYC$log_price)
     nbr.val     nbr.null       nbr.na          min          max        range
1.915900e+04 0.000000e+00 0.000000e+00 1.609438e+00 7.600402e+00 5.990964e+00
         sum       median         mean      SE.mean CI.mean.0.95          var
9.038740e+04 4.653960e+00 4.717752e+00 4.682934e-03 9.178962e-03 4.201544e-01
     std.dev     coef.var
6.481932e-01 1.373945e-01
```

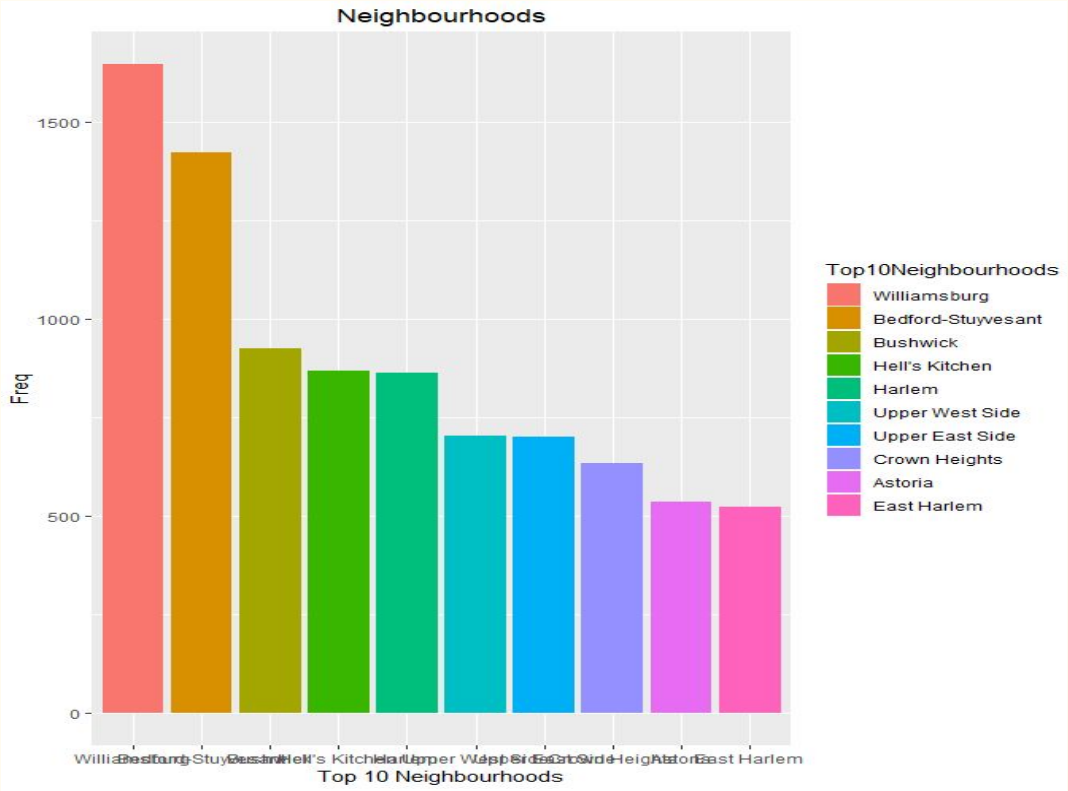
# Data Preparation & Exploration

- Data Visualization
  - Boxplot
  - Barplot
  - Violinplot
  - Histogram



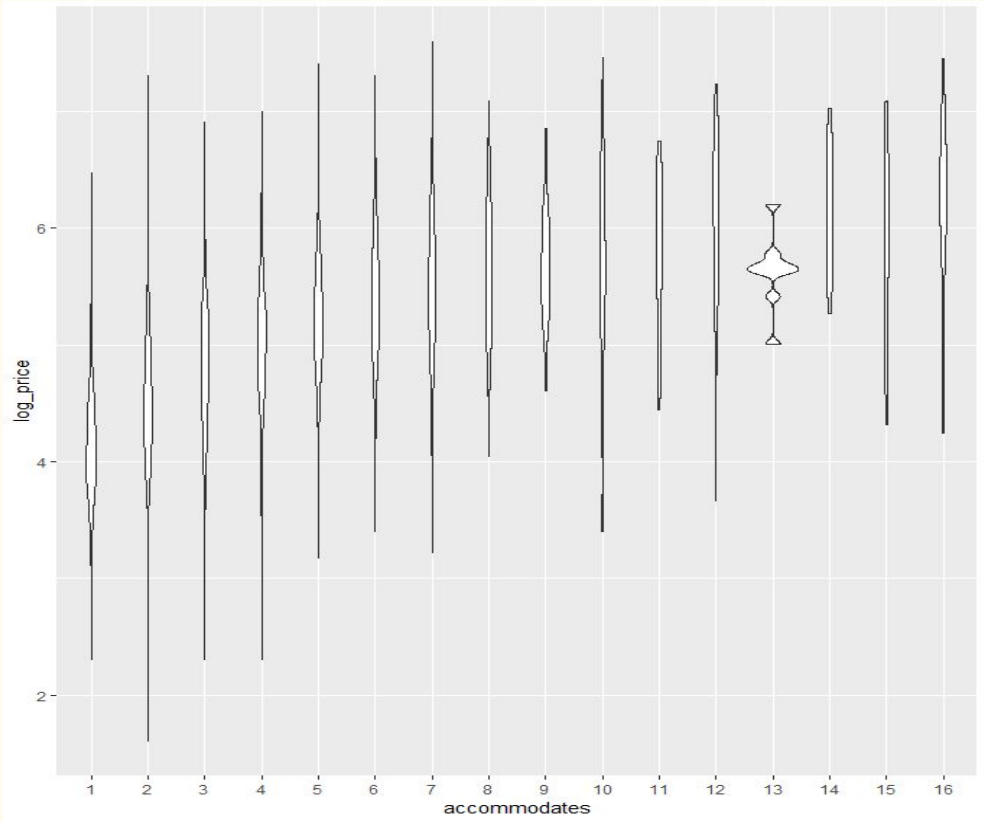
# Data Preparation & Exploration

- Data Visualization
  - Boxplot
  - Barplot
  - Violinplot
  - Histogram



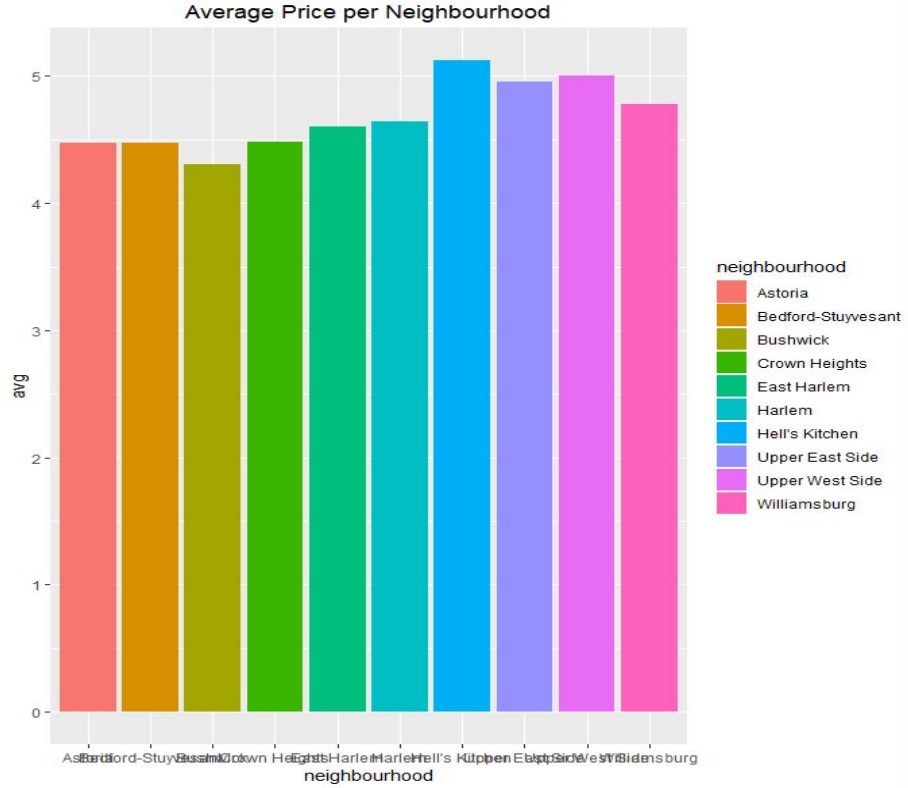
# Data Preparation & Exploration

- Data Visualization
  - Boxplot
  - Barplot
  - Violinplot
  - Histogram



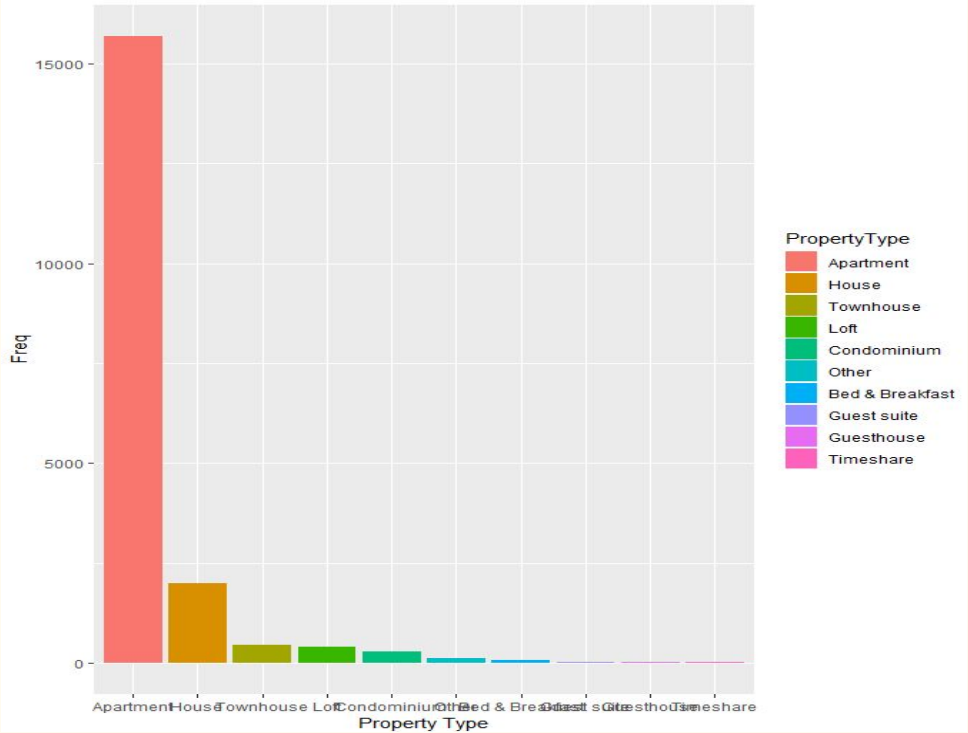
# Data Preparation & Exploration

- Data Visualization
  - Boxplot
  - Barplot
  - Violinplot
  - Histogram



# Data Preparation & Exploration

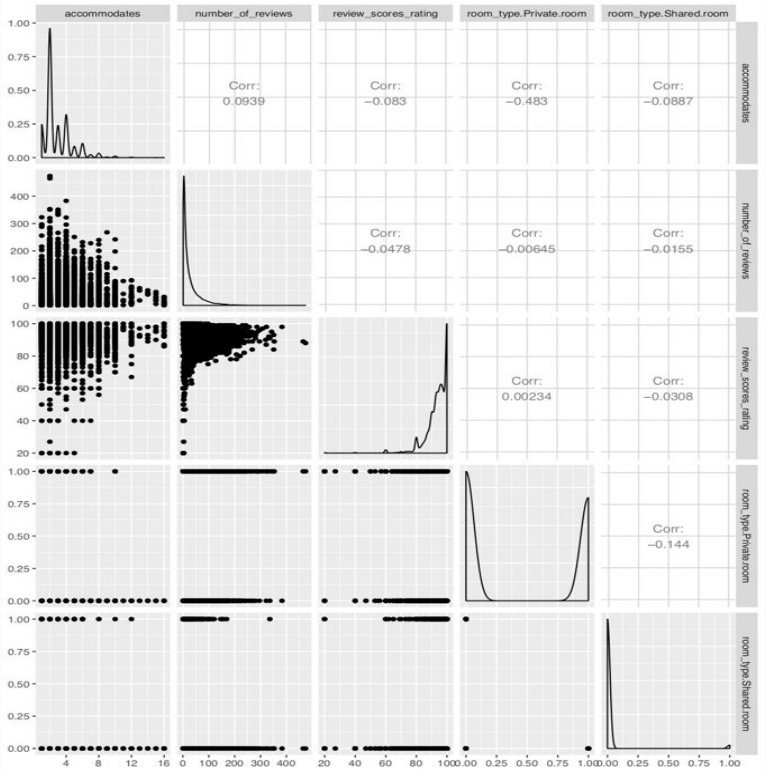
- Data Visualization
  - Boxplot
  - Barplot
  - Violinplot
  - Histogram



# Prediction

- Goals: Predict log_price
- Preparation for
  Multiple Regression Model
  - Property_type = Apartment
  - ggpair (avoid multicollinearity)
  - Dummy variables: room_type
  - Backward elimination
- Independent Variables
  - Accommodates, Review score rating, and Room types


ggpair with bathrooms, bedrooms, beds




ggpair of the independent variables

# Prediction

- The regression equation

Log_price = 4.2 + 0.11(accommodates)
+ 0.006(review_scores_rating)
– 0.61(room_type.Private.room)
– 0.95(room_type.Shared.room)

- R-square = 0.5343
  - Higher than the adjusted R-square
- RMSE (training) = 0.418
- RMSE (validation) = 0.423

```
> summary(fitall1)

Call:
lm(formula = log_price ~ accommodates + review_scores_rating +
    room_type.Private.room + room_type.Shared.room, data = training)
```

Model summary

```
Residual standard error: 0.4173 on 9571 degrees of freedom
Multiple R-squared:  0.5343,    Adjusted R-squared:  0.5341
F-statistic:  2745 on 4 and 9571 DF,  p-value: < 2.2e-16
```

R-square

```
> accuracy(pred1, training$log_price)
                     ME      RMSE       MAE        MPE     MAPE
Test set 1.216056e-13 0.4171511 0.3276648 -0.7807291 7.002059
> pred2 = predict(fitall1, validation)
> accuracy(pred2, validation$log_price)
                    ME      RMSE       MAE       MPE    MAPE
Test set -0.01062954 0.4231333 0.3260941 -1.064995 7.04626
```

Training set VS Validation set

# Classification

- **K-NN**
  - Outcome Variable
    - Cleaning Fee
  - Input Variables
    - Log_price
    - Accommodates
    - review score rating
  - Normalization
  - K-value
    - Test 98 neighbors
    - 
```
> sqrt(9576) #98 neighbourhoods for analysis
[1] 97.85704
```
    - Optimal value k=92
      - Highest Accuracy

```
90 90 0.7976775
91 91 0.7978410
92 92 0.7981681
93 93 0.7978410
94 94 0.7978410
95 95 0.7978410
96 96 0.7978410
97 97 0.7978410
98 98 0.7978410
```

```
        [,67]    [,68]    [,69]    [,70]    [,71]
[1,] 5.370127 5.387727 5.399269 5.399269 5.451011
       [,78]    [,79]    [,80]    [,81]    [,82]
[1,] 5.59162 5.624128 5.636191 5.664128 5.727984
        [,89]    [,90]    [,91]    [,92]
[1,] 5.794943 5.820293 5.826595 5.841224
Levels: True
```

# Classification

- **Naive-Bayes**
  - Cut into 4 different bins
    - "Student Budget" → [1.609 - 4.29]
    - "Below Average" → [4.29 - 4.733]
    - "Above Average" → [4.733 - 5.165]
    - "Pricey Digs" → [5.165 - 7.409]
  - Predictors
    - Accommodates, review score rating
    - Number of reviews, room type
  - Naive Rule → Below Average
    - 

```
Student Budget  Below Average  Above Average    Pricey Digs
          3192           4551           3631           4315
> 4551/15690
[1] 0.2900574
```

**Naive Bayes Records Classification**

| Student Budget | Below Average | Above Average | Pricey Digs |
|---|---|---|---|
| 0.1978967 | 0.2975356 | 0.2256214 | 0.2789463 |

**Training Matrix**

```
Confusion Matrix and Statistics

                Reference
Prediction       Student Budget Below Average Above Average Pricey Digs
  Student Budget            299           153            38          8
  Below Average            1484          1795           470        104
  Above Average              11            33            51         46
  Pricey Digs                69           820          1565       2468

Overall Statistics

               Accuracy : 0.49
                 95% CI : (0.4799, 0.5002)
    No Information Rate : 0.2975
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2902
 Mcnemar's Test P-Value : < 2.2e-16
```

**Validation Matrix**

```
Confusion Matrix and Statistics

                Reference
Prediction       Student Budget Below Average Above Average Pricey Digs
  Student Budget            216            90            19          6
  Below Average            1056          1117           327         83
  Above Average               8            14            30         30
  Pricey Digs                49           529          1131       1570

Overall Statistics

               Accuracy : 0.4674
                 95% CI : (0.455, 0.4798)
    No Information Rate : 0.2789
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.2707
 Mcnemar's Test P-Value : < 2.2e-16
```

# Classification Tree

Assess the Pruned tree:

- Cross-validation table
- Optimal Split

```
196 pruned.ct <- prune(df.cv,cp=df.cv$cptable[which.min(df.cv$cptable[,'xerror']),'CP'])
197 length(pruned.ct$frame$var[pruned.ct$frame$var=='<leaf>'])
198 rpart.plot(pruned.ct,type=4,extra=101,split.font=50,varlen=-10)
```

# Classification Tree

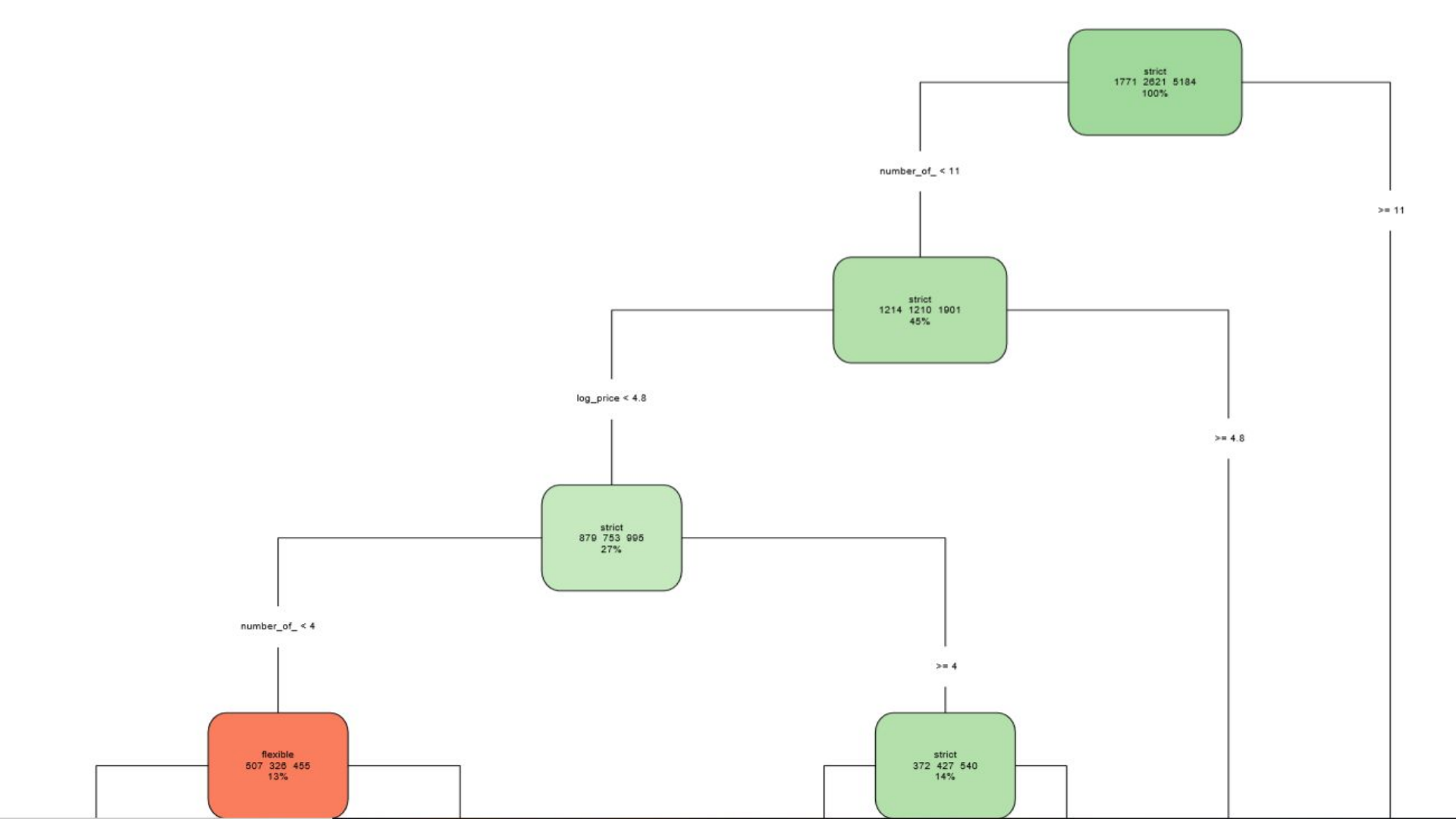
Classification tree: Prune the tree

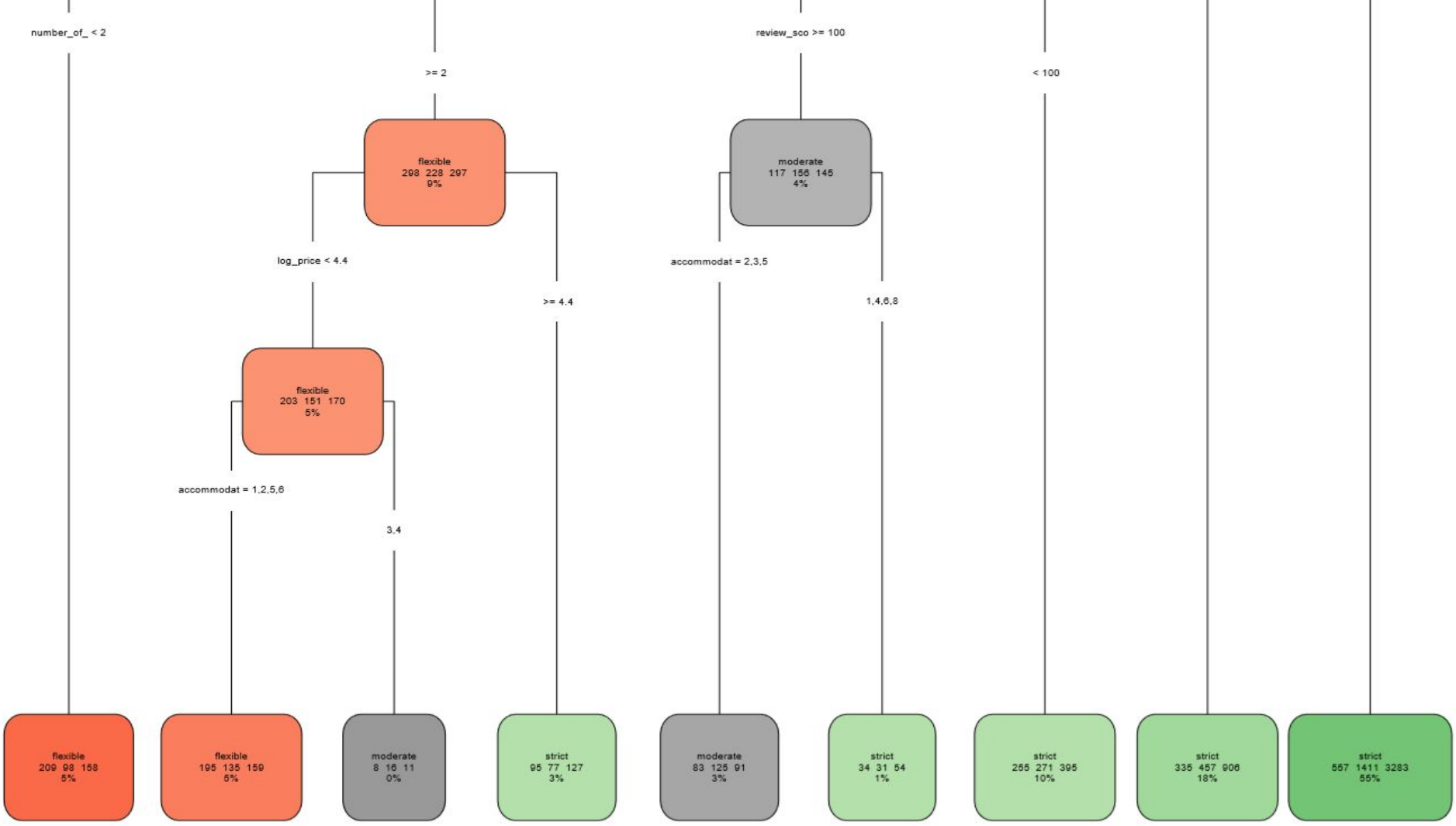
```
192 #Cross-validation
193 df.cv <- rpart(cancellation_policy~., data=trainingTree, method='class',
194                cp=0.00001,minsplit=5)
195 printcp(df.cv)
```

| | CP | nsplit | rel error | xerror | xstd |
|---|---|---|---|---|---|
| 1 | 3.9466e-03 | 0 | 1.00000 | 1.00000 | 0.011102 |
| 2 | 3.6430e-03 | 3 | 0.98816 | 1.00273 | 0.011104 |
| 3 | 2.5046e-03 | 5 | 0.98087 | 1.00296 | 0.011105 |
| 4 | 1.9353e-03 | 6 | 0.97837 | 0.99522 | 0.011098 |
| 5 | 1.3661e-03 | 8 | 0.97450 | 0.99317 | 0.011096 |
| 6 | 8.1967e-04 | 12 | 0.96903 | 0.99158 | 0.011095 |
| 7 | 7.9690e-04 | 17 | 0.96494 | 0.99294 | 0.011096 |
| 8 | 7.5896e-04 | 19 | 0.96334 | 0.99454 | 0.011097 |
| 9 | 6.8306e-04 | 32 | 0.95310 | 0.99772 | 0.011100 |

# Classification Tree

This is our 13-split classification tree:


strict
1771 2621 5184
100%
number_of_ < 11
>= 11
strict
1214 1210 1901
45%
log_price < 4.8
>= 4.8
strict
879 753 995
27%
number_of_ < 4
>= 4
flexible
507 328 455
13%
strict
372 427 540
14%


number_of_ < 2
>= 2
review_sco >= 100
< 100
flexible
298 228 297
9%
moderate
117 156 145
4%
log_price < 4.4
>= 4.4
accommodat = 2,3,5
1,4,6,8
flexible
203 151 170
5%
accommodat = 1,2,5,6
3,4
flexible
209 98 158
5%
flexible
195 135 159
5%
moderate
8 16 11
0%
strict
95 77 127
3%
moderate
83 125 91
3%
strict
34 31 54
1%
strict
255 271 395
10%
strict
335 457 906
18%
strict
557 1411 3283
55%

# Classification Tree

Assess the Pruned tree: Confusion Matrix

```
> confusionMatrix(df.pred,validationTree$cancellation_policy)
Confusion Matrix and Statistics

          Reference
Prediction flexible moderate strict
  flexible      273      158    258
  moderate       54       70     90
  strict        828     1393   2988

Overall Statistics

               Accuracy : 0.545
                 95% CI : (0.5324, 0.5575)
    No Information Rate : 0.5458
    P-Value [Acc > NIR] : 0.5563

                  Kappa : 0.0976
 Mcnemar's Test P-Value : <2e-16
```

```
> confusionMatrix(df.pred1,trainingTree$cancellation_policy)
Confusion Matrix and Statistics

          Reference
Prediction flexible moderate strict
  flexible      478      281    379
  moderate       89      145    108
  strict       1204     2195   4697

Overall Statistics

               Accuracy : 0.5556
                 95% CI : (0.5455, 0.5655)
    No Information Rate : 0.5414
    P-Value [Acc > NIR] : 0.00271

                  Kappa : 0.1295
 Mcnemar's Test P-Value : < 2e-16
```

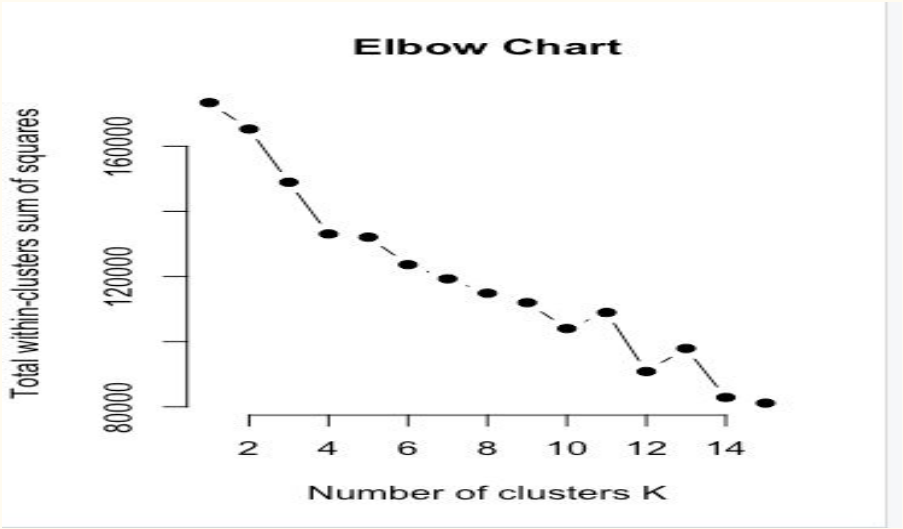
# Clustering

- To cluster the data we decided to choose the top 10 neighborhood in NYC based on their frequency. "Williamsburg", "Bedford-Stuyvesant", "Bushwick", "Hell's Kitchen", "Harlem","Upper West Side", "Upper East Side", "Crown Heights", "Astoria", "East Harlem".
- The variables that were considered for the clustering were log_price, accommodates, number_of_review, reviews_score_rating, bedrooms, beds, room type, neighbourhood, cancellation_policy and cleaning_fee.
- Super strict cancellation policy was disregarded.
- Neighbourhood, Cancellation Policy and cleaning fee were dummified

| flexible | moderate | strict | super_strict_30 | super_strict_60 |
|---|---|---|---|---|
| 1416 | 2049 | 4072 | 2 | 0 |

# Clustering

- The variables were normalized in order to make the comparison easier for each cluster
- Optimal number of cluster was identified based on the elbow chart
- Subjective choice



1. Manhattan Lovers
2. Picky Travellers
3. Organized Travelers
4. Families
5. Students
6. Flexible Travelers
7. Moderate Cancellation

```
> km1$centers
  log_price accommodates number_of_reviews review_scores_rating   bedrooms        beds room_type.Private.room
1 0.3794967  -0.14652759       -0.11357328          -0.058101448 -0.4373515 -0.15221059            -0.36022215
2 0.6599076  -0.01051254        0.26162541          -0.064496622 -0.1315443 -0.05737016            -0.18613125
3 -0.2975553 -0.30829526       -0.01533138           0.032229202 -0.2893255 -0.31749090             0.21641332
4 1.1950239   1.90104869        0.17827442          -0.149109749  1.8030294  1.87139475            -0.94545882
5 -0.5369664 -0.45352641       -0.12304015          -0.001062009 -0.2204836 -0.35242002             0.44453719
6 -0.4311341 -0.18212701       -0.08624767           0.116094093 -0.1073017 -0.18261185             0.12129149
7 -0.2058212 -0.21363531       -0.05288686           0.074241954 -0.2060289 -0.23698270             0.08977342
  room_type.Shared.room neighbourhood.Astoria neighbourhood.Bedford.Stuyvesant neighbourhood.Bushwick
1          0.103830001           -0.2437942                     -0.40866332           -0.33087422
2         -0.005992272           -0.2437942                     -0.40866332           -0.33087422
3         -0.013197734           -0.2437942                      0.09263021            0.07745046
4         -0.140058395           -0.1293248                      0.27461253           -0.06957163
5          0.128303931           -0.2437942                      0.04938613            0.28297204
6          0.011372841            4.1012759                     -0.40866332           -0.33087422
7         -0.041073524           -0.2437942                      0.13443416            0.09335380
  neighbourhood.Crown.Heights neighbourhood.East.Harlem neighbourhood.Harlem neighbourhood.Hell.s.Kitchen
1                -0.26650215                -0.26306057         -0.333090123                  -0.3500385
2                -0.26650215                -0.26306057         -0.333090123                   2.8564500
3                 0.05445496                 0.09207414          0.141626231                  -0.3500385
4                 0.05108182                 0.02244282         -0.003628636                  -0.1142155
5                 0.08733417                 0.11959674          0.108847212                  -0.2298301
6                -0.26650215                -0.26306057         -0.333090123                  -0.3500385
7                 0.14353044                 0.05210386          0.077985175                  -0.3500385
  neighbourhood.Upper.East.Side neighbourhood.Upper.West.Side neighbourhood.Williamsburg cancellation_policy.flexible
1                     3.2016994                   -0.31254842                -0.47499940                  -0.08538668
2                    -0.3122927                   -0.31254842                -0.47499940                  -0.24746540
3                    -0.3122927                    0.13039717                 0.21360895                   0.11236346
4                    -0.1772875                    0.11917755                -0.02753751                  -0.25603820
5                    -0.2510200                    0.01810184                 0.02210338                   0.64186883
6                    -0.3122927                   -0.31254842                -0.47499940                   0.07737335
7                    -0.3122927                    0.06584744                 0.22844166                  -0.48086226
  cancellation_policy.moderate cancellation_policy.strict cleaning_fee.False cleaning_fee.True
1                -0.084887355                 0.14270343        -0.0719403        0.0719403
2                -0.154059938                 0.33147929        -0.2536520        0.2536520
3                -0.611084840                 0.45751618        -0.4883744        0.4883744
4                -0.324930061                 0.49074978        -0.3853871        0.3853871
5                -0.005665272                -0.49796590         2.0473380       -2.0473380
6                 0.014104372                -0.07322876         0.1534977       -0.1534977
7                 1.636216922                -1.08396216        -0.4883744        0.4883744
```

# Conclusion

- Clustering helps Airbnb to identify different type of customers with different characteristics
- Methods like Naive Bayes, K-NN for outcome classification
- Training and validation for predictive purposes
- Understanding average prices per neighbourhood
- Useful to predict prices of new listings based on its characteristics

# Q & A