Airbnb Analysis

- based on Southbank Area, Melbourne

AD 699 Data Mining for Business Analytics
Instructor: Greg Page
Presented by: Dandan Zeng Haoran Zhang
Kaming Yip Yueyue Li Alan Lee
Date: December 10, 2019

Agenda

Project Overview

- Neighborhood Introduction
- Project Goal and Objectives

Data Preparation & Exploration

- Missing Values
- Visualization

Predication

- Multiple Regression

Classification

- K- nearest neighbors
- Naive Bayes
- Classification Tree

Clustering

- K-means Analysis

Conclusion

- Overview
- Outlook

Project Overview

Neighborhood Introduction

“Southbank is an inner urban neighbourhood of Melbourne, Victoria, Australia. Its local government area are the cities of Melbourne and Port Phillip. Its southernmost area is considered part of the **CBD** of the city. Southbank **Promenade** and Southgate **Restaurant** and **Shopping** Precinct, on the southern bank of the **Yarra River**, extending to **Crown Casino**, is one of Melbourne's major entertainment precincts.”

---------- Wikipedia

1

Data Preparation & Exploration

Missing Values

Method 1: Drop the N/A rows

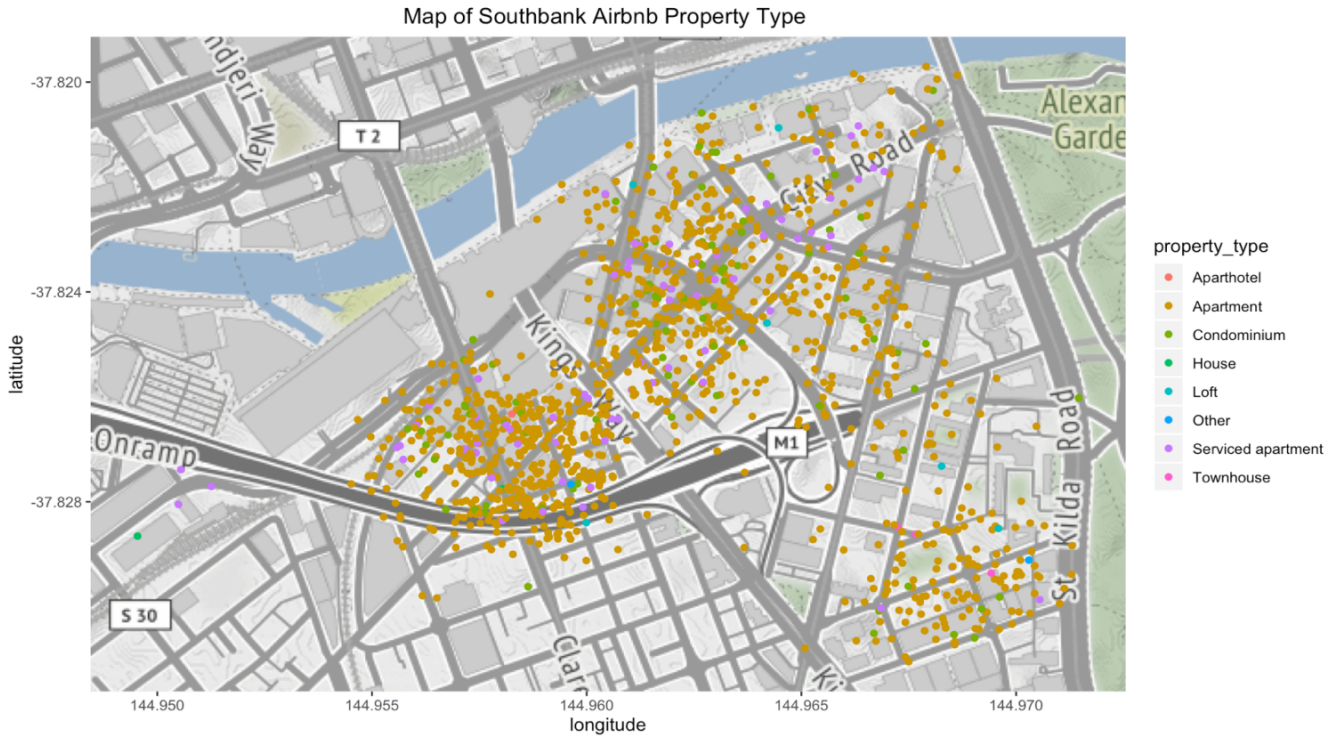
```
# Drop the rows
Southbank <- filter(Southbank, host_response_time != "NA", host_response_rate != "NA",
                    host_is_superhost != "NA", host_identity_verified != "NA")
```

Method 2: Imputation

- Security_deposit ------ “0”
- Cleaning_fee ------ “0”
- Review_score_rating ------ “median”
- Review_score_accuracy ------ “median”

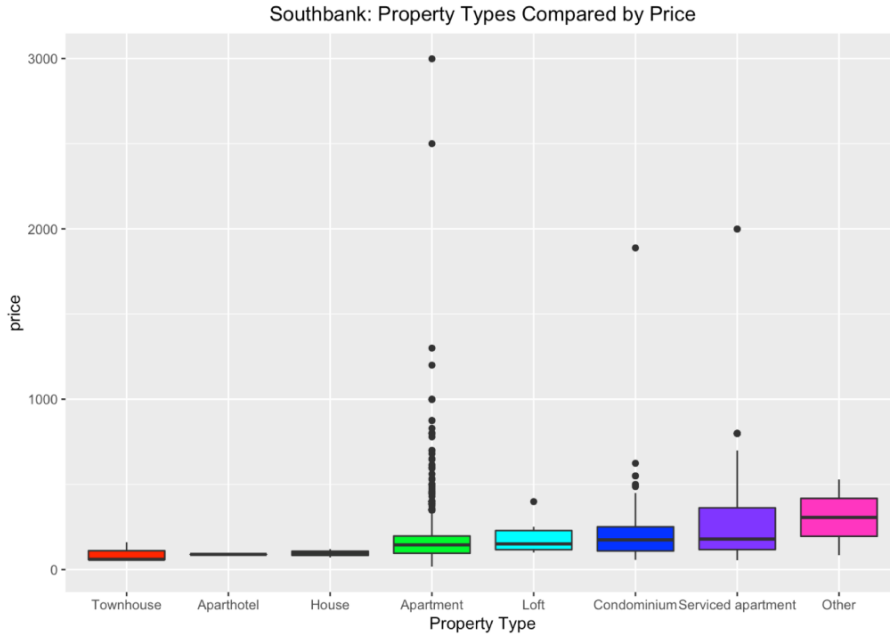
Visualization

Plot 1: map

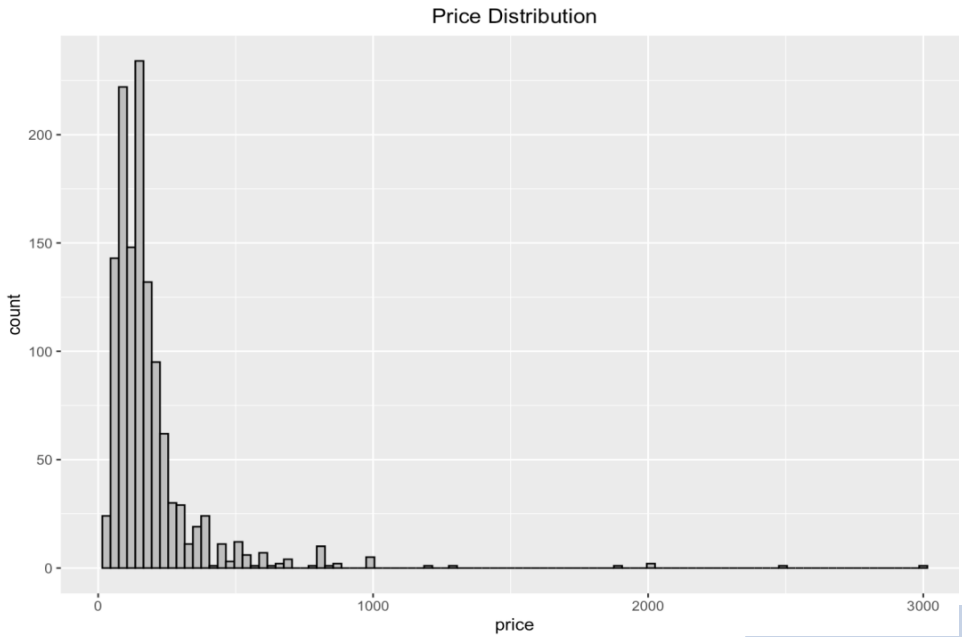


Visualization

Plot 2: Boxplot

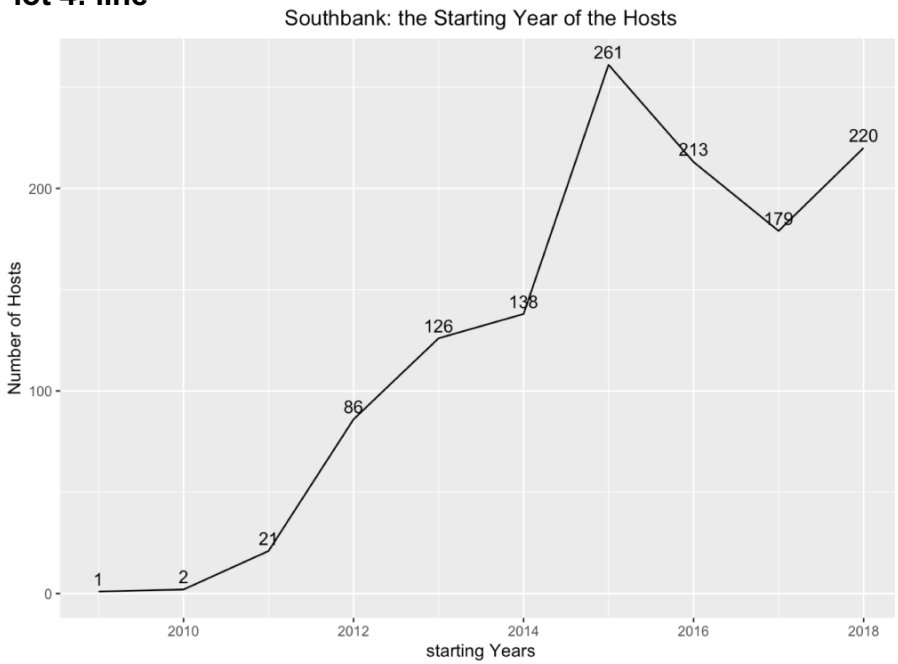


Plot 3: Histogram

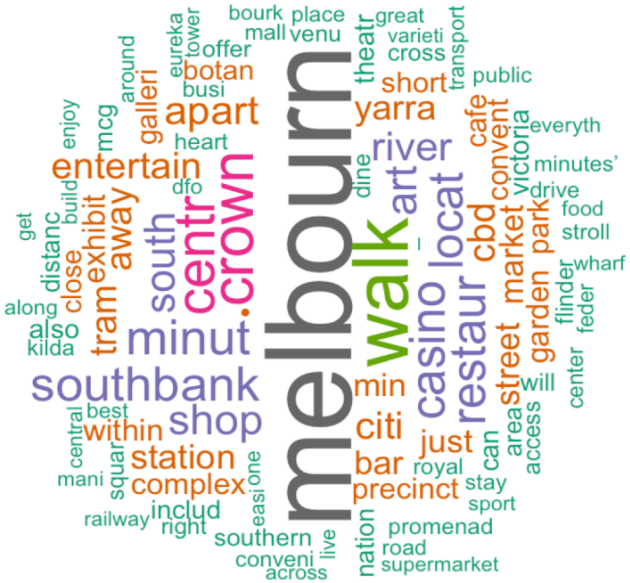


Visualization

Plot 4: line



Plot 5: word cloud



2

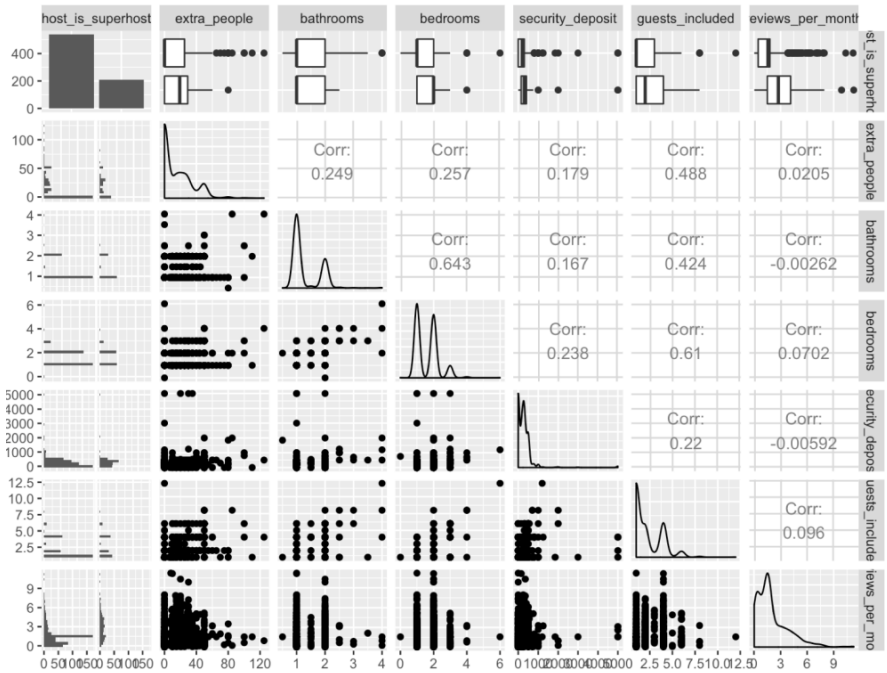
Prediction

Prediction
- Multiple Linear Regression

Selecting Significant Predictors
Backward elimination

Final Predictors

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       -68.77898   21.81436  -3.153 0.001682 **
host_is_superhostt -54.88633   16.49717  -3.327 0.000921 ***
bathrooms          99.56414   18.30919   5.438 7.33e-08 ***
bedrooms           52.02122   15.41652   3.374 0.000778 ***
security_deposit    0.11245    0.01665   6.755 2.90e-11 ***
guests_included    14.24800    6.33968   2.247 0.024906 *
reviews_per_month  -8.34772    4.13917  -2.017 0.044081 *
extra_people        1.24841    0.41736   2.991 0.002871 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 190.5 on 740 degrees of freedom
Multiple R-squared:  0.3053,    Adjusted R-squared:  0.2988
F-statistic: 46.46 on 7 and 740 DF,  p-value: < 2.2e-16
```



Prediction
- Multiple Linear Regression

Regression Equation

*Price = -68.78 – 54.89*host_is_superhostt + 99.56*bathrooms + 52.02*bedrooms +*

*0.11*security_deposit + 14.29*guests_included – 8.35*review_per_month + 1.25*extra_people*

Accuracy

```
> ## Prediction accuracy
> pred_train <- predict(Southbank_MLR_final, Southbank_MLR_train)
> accuracy(pred_train, Southbank_MLR_train$price)  #accuracy against trainset
                     ME     RMSE      MAE       MPE     MAPE
Test set -3.184941e-12 189.4599 89.02851 -24.43448 51.39792
> pred_valid <- predict(Southbank_MLR_final, Southbank_MLR_valid)  #accuracy against validset
> accuracy(pred_valid, Southbank_MLR_valid$price)
                ME     RMSE      MAE       MPE     MAPE
Test set -18.74549 129.9477 82.94754 -31.81888 57.28158
```

3

Classification

Classification
- k-nearest Neighbors

Predict Cancellation Policy

```
> levels(KNN$cancellation_policy)
[1] "flexible"                "moderate"                "strict_14_with_grace_period"
[4] "super_strict_30"         "super_strict_60"
```

- Predictors:
  - The capacity of rooms: room types; space available
  - The price: direct influence on cancellation policy
  - The host attitude: standard; bookable time available

```
> str(KNN)
'data.frame':    1247 ol
```

Room Types
- $ accommodates
- $ bathrooms
- $ bedrooms
- $ beds

Price
- $ price
- $ security_deposit
- $ cleaning_fee

Space Available
- $ guests_included
- $ extra_people

Standard
- $ minimum_nights
- $ maximum_nights

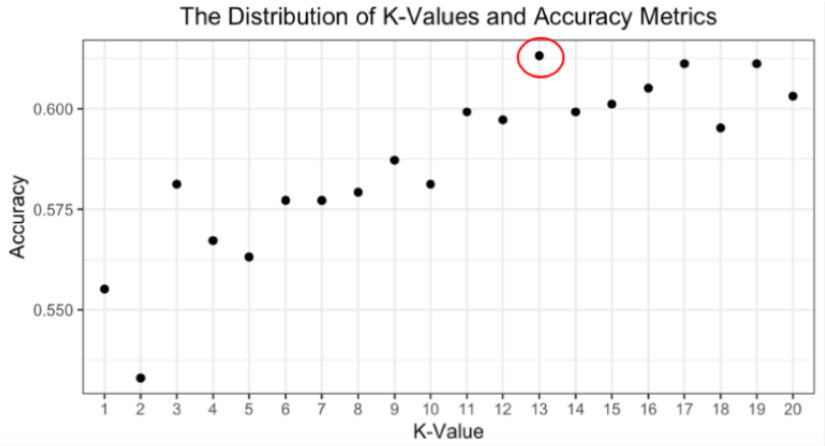
Bookable Time Available
- $ availability_30
- $ availability_60
- $ availability_90
- $ availability_365

- $ cancellation_policy

Classification
- k-nearest Neighbors

- **Optimal K-Value**
  - K = 13
- **Highest Accuracy**
  - 61.32%

```
> accuracy.df
    k  accuracy
1   1 0.5551102
2   2 0.5330661
3   3 0.5811623
4   4 0.5671343
5   5 0.5631263
6   6 0.5771543
7   7 0.5771543
8   8 0.5791583
9   9 0.5871743
10 10 0.5811623
11 11 0.5991984
12 12 0.5971944
13 13 0.6132265
14 14 0.5991984
15 15 0.6012024
16 16 0.6052104
17 17 0.6112224
18 18 0.5951904
19 19 0.6112224
20 20 0.6032064
> accuracy.df %>% filter(accuracy==max(accuracy))
   k  accuracy
1 13 0.6132265
```



Classification
- k-nearest Neighbors

- **Create Test Neighborhood**

```
# create new data.frame
knn.new.data <- data.frame(accommodates=4, bathrooms=2, bedrooms=2, beds=3, price=200,
                           security_deposit=200, cleaning_fee=100, guests_included=2,
                           extra_people=0, minimum_nights=1,maximum_nights=365,
                           availability_30=7, availability_60=56, availability_90=77,
                           availability_365=215)
```

- **Test Neighborhood**

```
> nn
[1] strict_14_with_grace_period
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]  406  349  129  614  457  433  354  273  173   434   253   235
     [,13]
[1,]   618
attr(,"nn.dist")
         [,1]     [,2]     [,3]     [,4]    [,5]     [,6]     [,7]
[1,] 2.153631 2.213216 2.352035 2.482643 2.54372 2.612898 2.648683
         [,8]     [,9]    [,10]    [,11]    [,12]    [,13]
[1,] 2.659395 2.660477 2.675556 2.676061 2.686176 2.697955
Levels: strict_14_with_grace_period
```

Classification
- Naive Bayes

Data Preprocessing:

- **Bin the numerical predictors and convert them to categorical predictors**

```
> str(naive_Bayes)
'data.frame':   1247 obs. of  15 variables:
 $ host_since                      : Factor w/ 10 levels "2009","2010",..: 3 4 3 3 5 4 3 5 5 5 ...
 $ host_response_time              : Factor w/ 5 levels "a few days or more",..: 5 5 5 5 5 4 5 2 4 5 ...
 $ host_is_superhost               : Factor w/ 2 levels "f","t": 1 1 1 1 2 2 1 1 2 1 ...
 $ property_type                   : Factor w/ 8 levels "Aparthotel","Apartment",..: 7 2 7 2 2 2 2 2 2 3 ...
 $ room_type                       : Factor w/ 3 levels "Entire home/apt",..: 1 1 1 2 1 1 1 1 1 1 ...
 $ accommodates                    : Factor w/ 3 levels "Single or Couple Size",..: 3 2 2 1 2 1 3 2 3 2 ...
 $ price                           : Factor w/ 4 levels "Economic","Affordable",..: 4 2 4 1 3 2 4 2 4 4 ...
 $ security_deposit                : Factor w/ 3 levels "No Fee","Market Price",..: 2 2 2 2 2 2 2 2 2 3 ...
 $ cleaning_fee                    : Factor w/ 3 levels "No Fee","Market Price",..: 2 3 2 2 2 2 2 3 3 1 ...
 $ review_scores_rating            : Factor w/ 4 levels "Below Avg","General Taste",..: 3 4 3 4 4 4 3 4 4 4 ...
 $ review_scores_communication     : Factor w/ 4 levels "Below Avg","General",..: 3 4 4 3 4 4 3 3 3 3 ...
 $ cancellation_policy             : Factor w/ 5 levels "flexible","moderate",..: 3 3 3 3 2 2 3 3 3 3 ...
 $ require_guest_profile_picture   : Factor w/ 2 levels "f","t": 1 2 1 1 1 1 1 2 1 1 ...
 $ require_guest_phone_verification: Factor w/ 2 levels "f","t": 2 2 2 2 1 1 2 2 1 1 ...
 $ instant_bookable                : Factor w/ 2 levels "f","t": 2 2 2 1 2 1 2 1 1 2 ...
```

Classification
- Naive Bayes

After data partition:

- **Build a Naive Bayes Model against training data**

```
> naive_Bayes_Train_nb <- naiveBayes(instant_bookable ~ ., data = naive_Bayes_Train)
> naive_Bayes_Train_nb

Naive Bayes Classifier for Discrete Predictors

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

A-priori probabilities:
Y
        f         t
0.4237968 0.5762032

Conditional probabilities:
   host_since
Y           2009        2010        2011        2012        2013        2014        2015        2016        2017        2018
  f 0.003154574 0.003154574 0.015772871 0.059936909 0.145110410 0.145110410 0.230283912 0.173501577 0.110410095 0.113564669
  t 0.000000000 0.000000000 0.011600928 0.092807425 0.083526682 0.067285383 0.199535963 0.148491879 0.162412993 0.234338747
```

(partial result)

Classification
- Naive Bayes

Model Assessment: Confusion Matrix

```
Confusion Matrix and Statistics

          Reference
Prediction   f   t
         f 192 106
         t 125 325

               Accuracy : 0.6912
                 95% CI : (0.6567, 0.7241)
    No Information Rate : 0.5762
    P-Value [Acc > NIR] : 6.286e-11

                  Kappa : 0.3626

 Mcnemar's Test P-Value : 0.2363

            Sensitivity : 0.6057
            Specificity : 0.7541
         Pos Pred Value : 0.6443
         Neg Pred Value : 0.7222
             Prevalence : 0.4238
         Detection Rate : 0.2567
   Detection Prevalence : 0.3984
      Balanced Accuracy : 0.6799

       'Positive' Class : f
```

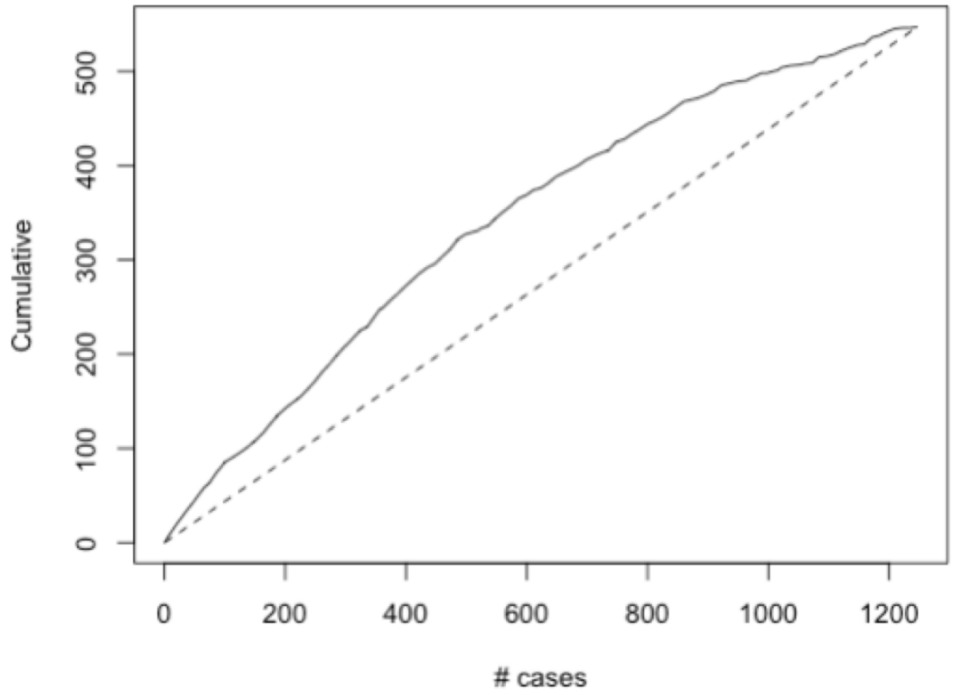
Training Data:
accuracy: 0.6912
sensitivity: 0.6057
Specificity: 0.7541
Pos Pred Value: 0.6443
Neg Pred Value:0.7222

Validation Data:
accuracy: 0.6974
sensitivity: 0.6087
Specificity: 0.7732
Pos Pred Value: 0.6965
Neg Pred Value:0.6980

```
Confusion Matrix and Statistics

          Reference
Prediction   f   t
         f 140  61
         t  90 208

               Accuracy : 0.6974
                 95% CI : (0.655, 0.7374)
    No Information Rate : 0.5391
    P-Value [Acc > NIR] : 3.726e-13

                  Kappa : 0.3855

 Mcnemar's Test P-Value : 0.02269

            Sensitivity : 0.6087
            Specificity : 0.7732
         Pos Pred Value : 0.6965
         Neg Pred Value : 0.6980
             Prevalence : 0.4609
         Detection Rate : 0.2806
   Detection Prevalence : 0.4028
      Balanced Accuracy : 0.6910

       'Positive' Class : f
```

Classification
- Naive Bayes

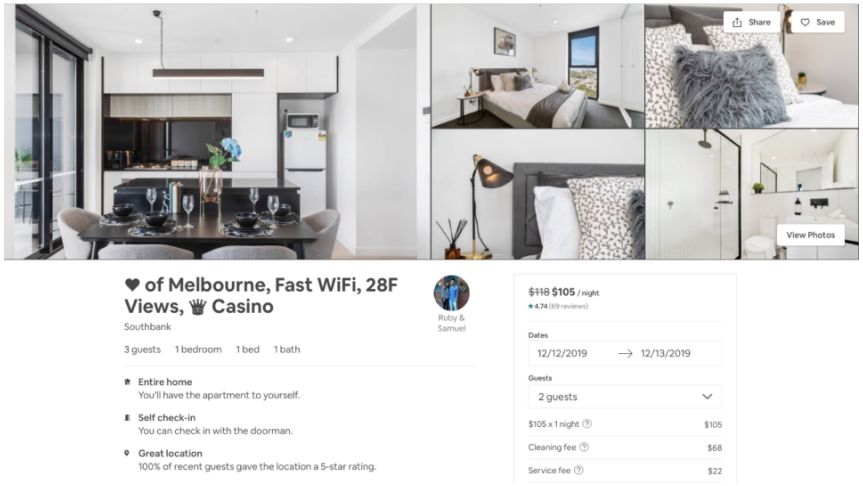
Model Assessment: Lift Chart



Classification
- Naive Bayes

Model Testing: a fictional apartment

```
> # Fictional Apartment
> Cozy_apt <- data.frame(host_since = "2018", host_response_time = "within an hour",
+                        host_is_superhost = "f", property_type = "Apartment",
+                        room_type = "Entire home/apt", accommodates = "Friends or Family Size",
+                        price = "Affordable", security_deposit = "Market Price",
+                        cleaning_fee = "High Fee", review_scores_rating = "Extraordinary",
+                        review_scores_communication = "Excellent", cancellation_policy = "moderate",
+                        require_guest_profile_picture = "f", require_guest_phone_verification = "f")
```



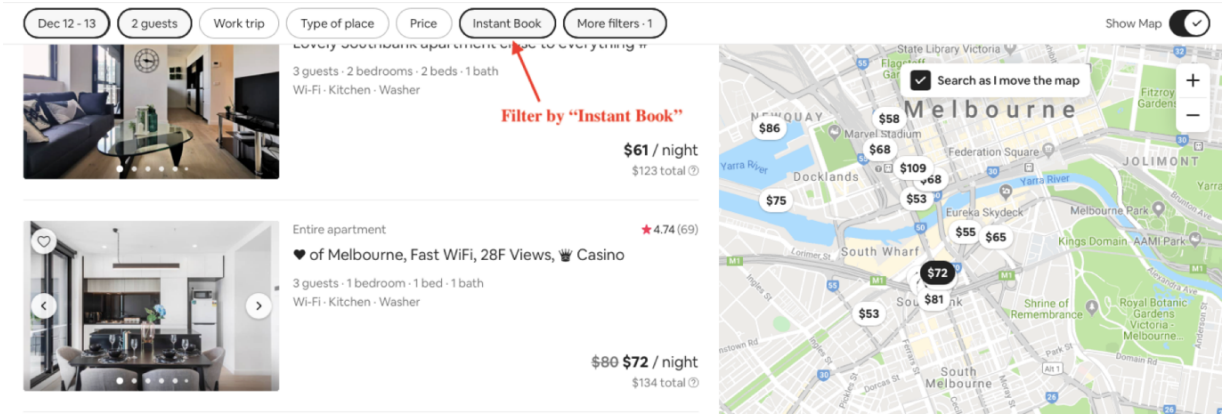
Classification
- Naive Bayes

Model Testing: a fictional apartment

```
> # predict probability
> Cozy_apt_pred_prob <- predict(naive_Bayes_nb, Cozy_apt, type = "raw")
> Cozy_apt_pred_prob
              f         t
[1,] 0.1151591 0.8848409
> # predict class membership
> Cozy_apt_pred_class <- predict(naive_Bayes_nb, Cozy_apt)
> Cozy_apt_pred_class
[1] t
Levels: f t
> cat("The Cozy Apt is predicted as", ifelse(Cozy_apt_pred_class == "t", "instantly bookable", "not instantly bookable"))
The Cozy Apt is predicted as instantly bookable
```

Happy to see the model gives out the same "decision" as the hosts Ruby & Samuel!

instantly bookable!



Classification
- Classification Tree

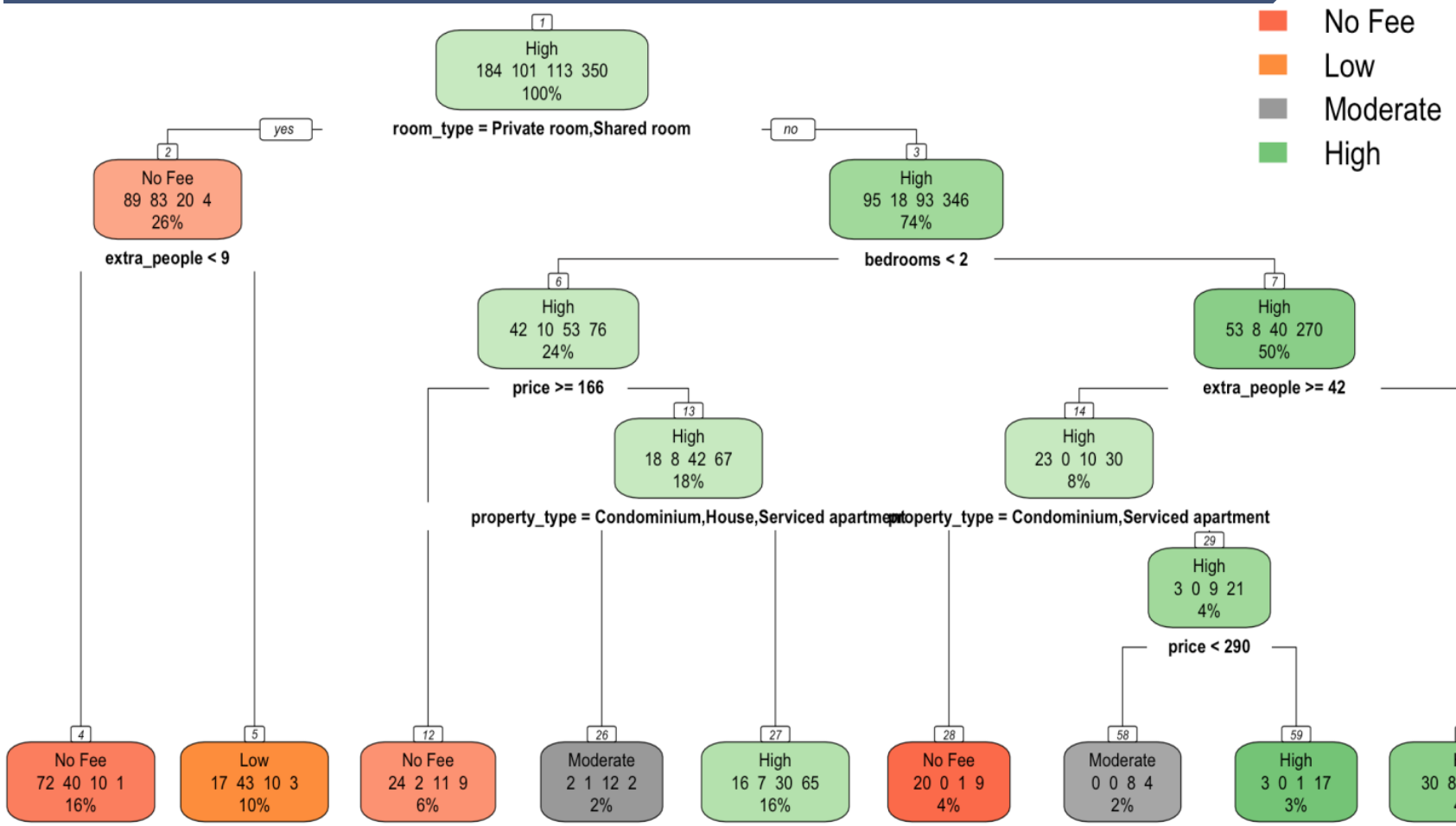
```
> summary(Southbank$cleaning_fee)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   10.00   60.00   60.18   95.50  450.00
```

```
Southbank$cleaning_fee2 <- cut(Southbank$cleaning_fee, breaks =
                        c(-0.1, 0.1, 35, 60,450), labels = c("No Fee", "Low","Moderate", "High"))
```

4 groups: No Fee (-0.1, 0.1)
Low (0.1, 35)
Moderate (35, 60)
High (60, 45)

Classification
- Classification Tree

- Classification Tree - First trail



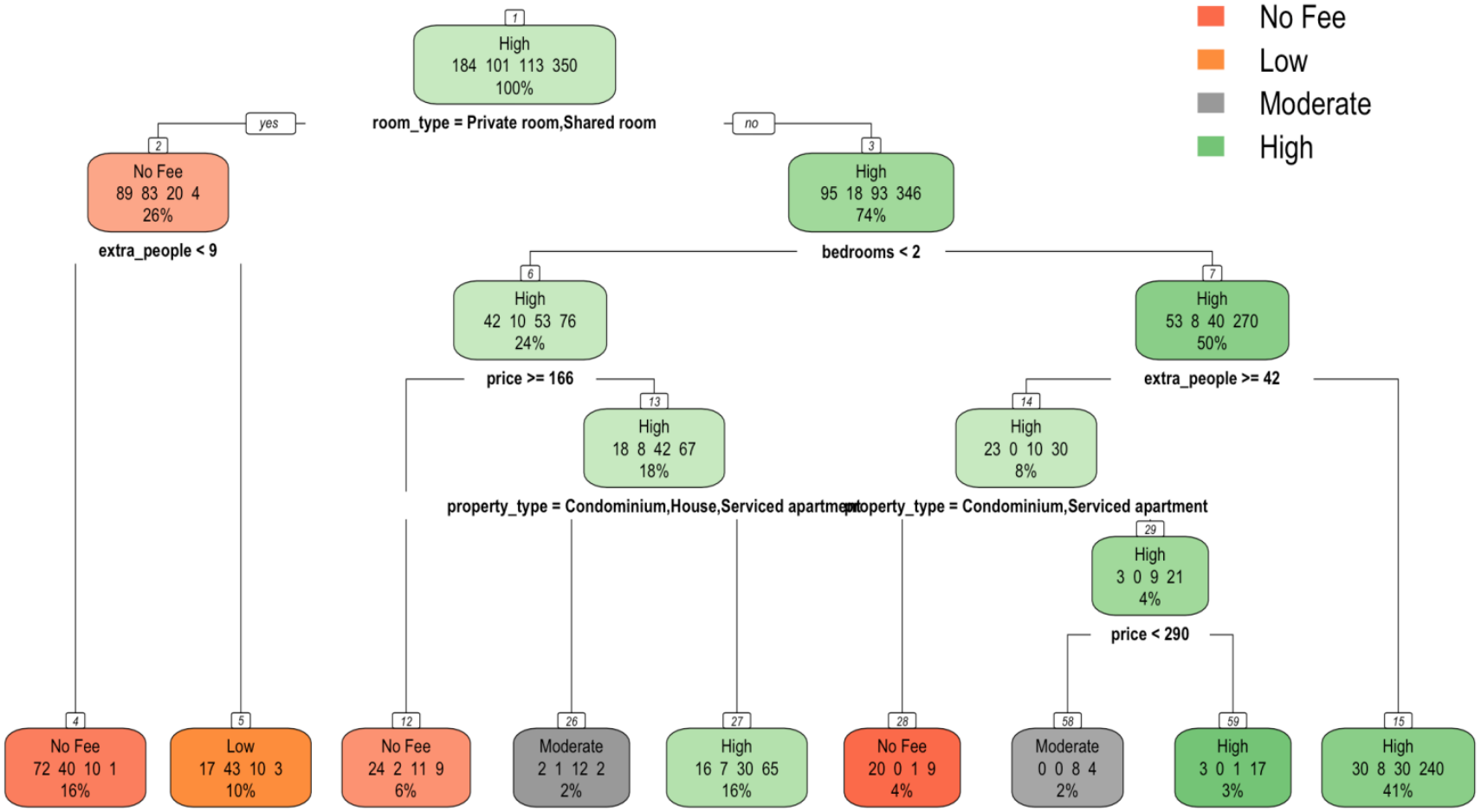
Classification
- Classification Tree

```
Classification tree:
rpart(formula = cleaning_fee2 ~ property_type + room_type + accommodates +
    bathrooms + bedrooms + beds + price + guests_included + extra_people +
    review_scores_cleanliness, data = train, method = "class",
    cp = 1e-05, xval = 5)

Variables actually used in tree construction:
[1] accommodates             bedrooms                 beds                     extra_people
[5] guests_included          price                    property_type            review_scores_cleanliness
[9] room_type

Root node error: 398/748 = 0.53209

n= 748

           CP nsplit rel error  xerror     xstd
1  0.21356784      0   1.00000 1.00000 0.034288
2  0.06532663      1   0.78643 0.80905 0.034025
3  0.01884422      2   0.72111 0.72362 0.033438
4  0.01381910      5   0.65829 0.70352 0.033256
5  0.01005025      7   0.63065 0.69598 0.033183
6  0.00502513      8   0.62060 0.72111 0.033416
7  0.00376884     14   0.58794 0.72111 0.033416
8  0.00335008     16   0.58040 0.71106 0.033326
9  0.00251256     20   0.56281 0.74121 0.033584
10 0.00167504     24   0.55276 0.73367 0.033523
11 0.00083752     27   0.54774 0.75628 0.033698
12 0.00001000     30   0.54523 0.76131 0.033734
```

cp=0.01005025

Classification

- Classification Tree - with ideal size



No Change

Classification
- Classification Tree - Assess the tree

```
> pred.train<-predict(treemodel3,newdata = train,type = "class")
> confusionMatrix(pred.train,train$cleaning_fee2)
Confusion Matrix and Statistics

          Reference
Prediction No Fee Low Moderate High
  No Fee      116  42       22   19
  Low          17  43       10    3
  Moderate      2   1       20    6
  High         49  15       61  322

Overall Statistics

               Accuracy : 0.6698
                 95% CI : (0.6348, 0.7034)
    No Information Rate : 0.4679
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4807

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

Statistics by Class:

                     Class: No Fee Class: Low Class: Moderate Class: High
Sensitivity                 0.6304    0.42574          0.17699      0.9200
Specificity                 0.8528    0.95363          0.98583      0.6859
Pos Pred Value              0.5829    0.58904          0.68966      0.7204
Neg Pred Value              0.8761    0.91407          0.87065      0.9070
Prevalence                  0.2460    0.13503          0.15107      0.4679
Detection Rate              0.1551    0.05749          0.02674      0.4305
Detection Prevalence        0.2660    0.09759          0.03877      0.5976
Balanced Accuracy           0.7416    0.68969          0.58141      0.8030
```

67%

```
> pred.valid<-predict(treemodel3,newdata = valid,type = "class")
> confusionMatrix(pred.valid,valid$cleaning_fee2)
Confusion Matrix and Statistics

          Reference
Prediction No Fee Low Moderate High
  No Fee       59  25       16   10
  Low           9  37        8    4
  Moderate      2   1        5   11
  High         45   7       44  216

Overall Statistics

               Accuracy : 0.6353
                 95% CI : (0.5913, 0.6776)
    No Information Rate : 0.483
    P-Value [Acc > NIR] : 5.522e-12

                  Kappa : 0.4168

 Mcnemar's Test P-Value : 1.890e-12

Statistics by Class:

                     Class: No Fee Class: Low Class: Moderate Class: High
Sensitivity                 0.5130    0.52857          0.06849      0.8963
Specificity                 0.8672    0.95105          0.96714      0.6279
Pos Pred Value              0.5364    0.63793          0.26316      0.6923
Neg Pred Value              0.8560    0.92517          0.85833      0.8663
Prevalence                  0.2305    0.14028          0.14629      0.4830
Detection Rate              0.1182    0.07415          0.01002      0.4329
Detection Prevalence        0.2204    0.11623          0.03808      0.6253
Balanced Accuracy           0.6901    0.73981          0.51781      0.7621
```

64%

4

Clustering

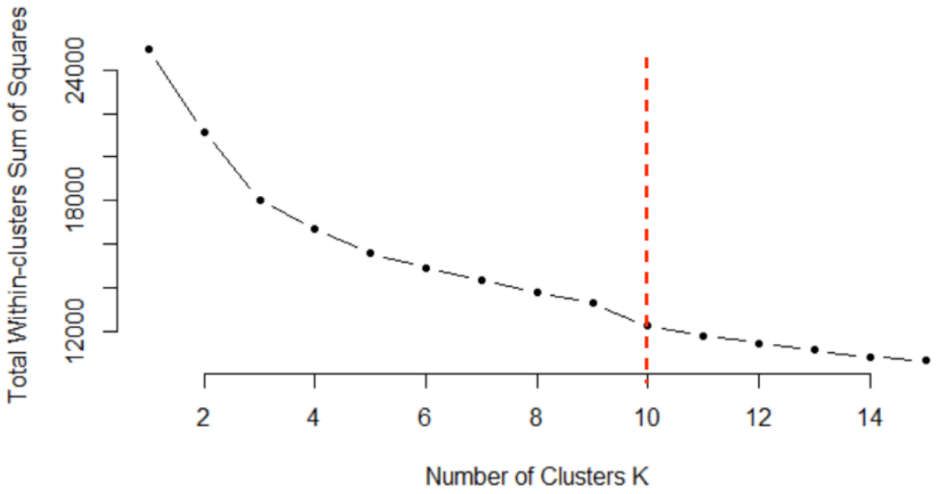
Clustering

Data Preprocessing:

- **Normalization on the selected numerical variables**

Selecting k clusters:

- **Elbow Chart**



Clustering

Fun Time:

- **Cluster 1: Families. High accommodates, bathrooms, bedrooms, beds**
- **Cluster 9: Herd. Even higher accommodates, bathrooms, bedrooms, beds**

```
> km$centers
   accommodates   bathrooms    bedrooms        beds       price security_deposit cleaning_fee
1    0.62029804  0.42249755  0.69552057   0.4443578  0.06680139     0.1474762987   0.56394304
2   -0.73627743 -0.47012065 -0.66283818  -0.6278049 -0.35607131    -0.3498698875  -0.42552654
3    0.58935120  0.67456791  0.72193436   0.5416088  1.60462103     0.0979084551  -0.41903663
4   -0.28234608 -0.26224231 -0.24257672  -0.3072391 -0.14127050     0.0882184476  -0.05928279
5   -1.39690083 -0.62308772 -2.30000128  -0.7848928 -0.77657956    -0.6906790127  -1.13857077
6   -0.73185773 -0.52935019 -0.80244336  -0.5983872 -0.34344272    -0.2869426988  -0.52605030
7    0.62200410  0.04986824  0.20317629   0.2990839 -0.13788788    -0.0006467323   0.12253105
8   -0.09582876 -0.22615777 -0.02260051  -0.2460015  3.12124747    11.2123778235   0.52637069
9    2.28716928  1.94326976  2.28915760   2.8038444  1.37582368     0.8993248950   1.62893773
10  -0.11012626  0.22747647 -0.06326838  -0.3999704  0.06906698     0.1225733177  -0.47908097
```

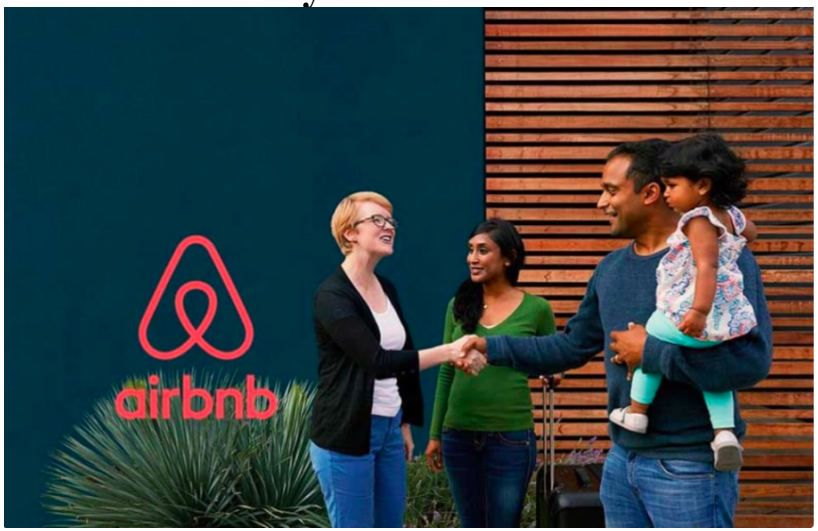
5

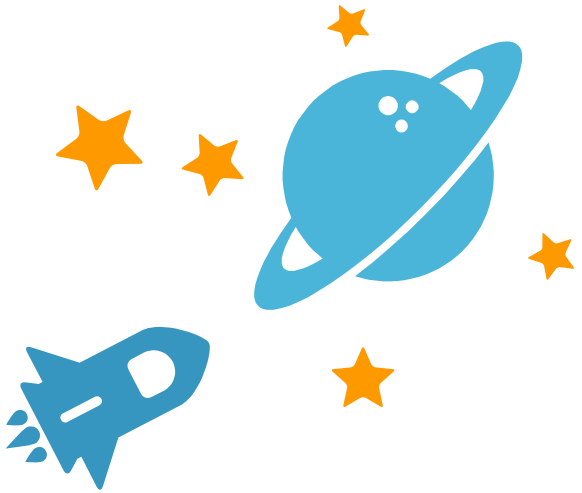
Conclusion

Conclusion

How can these models be useful? Who could benefit from these models?

- **For hosts, to pinpoint their Airbnb property features**
- **For Airbnb, to better classify their hosts**





THANK YOU

FOR YOUR PATIENCE!