AD699-Data Mining Team Presentation

By: Kalyan Iskepally, Neal Picard, Jose Martinez, Greg Parker

Table of Contents

- Data Preparation and Exploration
  - Missing Values
  - Summary Statistics
  - Visualization
- Prediction
  - Multiple Linear Regression
  - MLR Formula
  - MLR Summary
- Classification
  - K-Nearest Neighbors
  - Naive Bayes
  - Classification Tree
- Clustering
  - Feature Engineering
  - K-Means Clustering
- Conclusion

Missing Values

- Data Preparation
  - Filter imported .csv file to only include Boston
  - Replace NULL values with NA and the impute median values to replace NA's, remove unnecessary values
  - Selected variables were left at 0 with full filtered data at 3468 observations and 29 variables

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

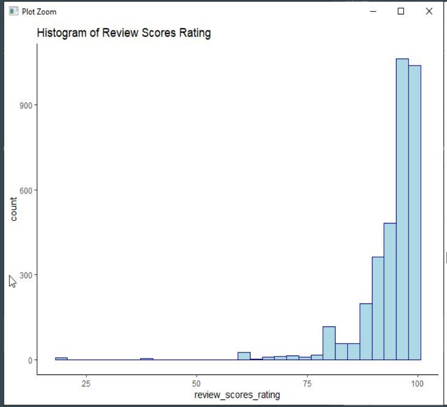
Summary of Statistics

- Summary Statistics
  - Utilizing the summary() function, we explored several variables
- Review score ratings:
  - Median = 96 and Mean = 94.05
  - Distribution -> Negatively Skewed
- Log price/Nightly price:
  - Median Log = 4.913 and Mean = 4.884
  - Distribution for Log price -> Negatively Skewed

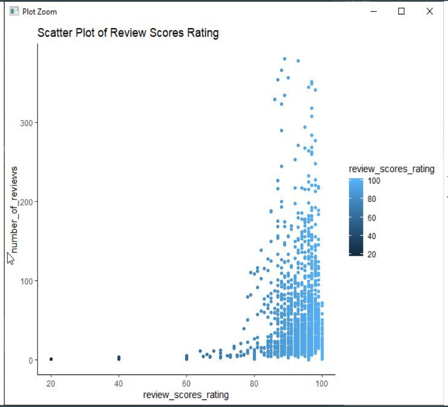
```
summary(boston1$review_scores_rating)

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

summary(boston1$log_price)

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

summary(boston1$nightly_price)

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

sd(boston1$review_scores_rating)

[1] 7.327312

sd(boston1$log_price)

[1] 0.6646924

sd(boston1$nightly_price)

[1] 128.8892
```

Visualization

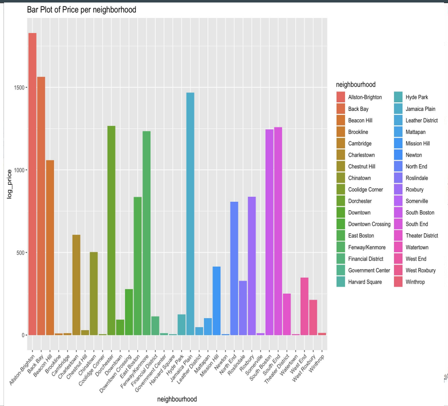
➔ Visualization
- Utilizing the ggplot2() package, we created several summary visualizations that helped us explore the data



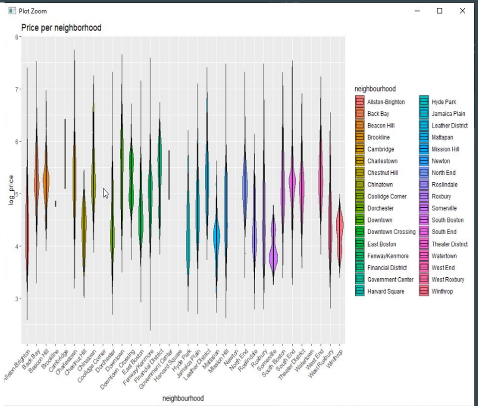
geom_hist()



geom_point()



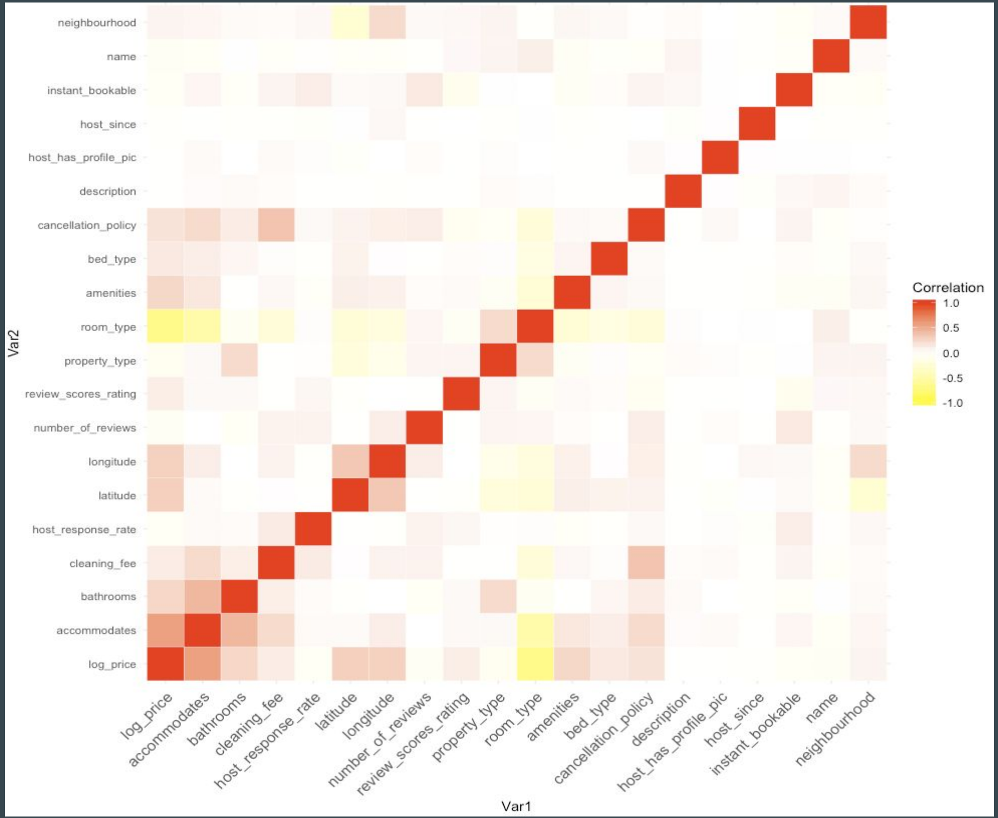
geom_bar()



geom_violin()

Multiple Linear Regression

- Selected Variables
  - Include all variables but eliminate variables that have a strong correlation
    - Prevent multicollinearity
- Eliminated variables
  - ID - not necessary
  - Nightly price
    - This was the same as log price but preserved at a true dollar format
  - Beds and Bedroom
    - Both were correlated with accommodates
- Finalized Variables
  - Per the following heat map, our aim was to eliminate anything in bright red.
  - No bright red was shown in the heat map, therefore we left as is



Multiple Linear Regression - Regression Formula

```
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    host_response_rate
          1.20057241            0.08197770            0.12304396           -0.09051680           -0.20288957
            latitude             longitude     number_of_reviews  review_scores_rating         property_type
          3.82431067            2.22496573           -0.00046209            0.00401201            0.00244805
           room_type             amenities              bed_type      instant_bookable         neighbourhood
         -0.61106801            0.00000178            0.04922103           -0.06601924            0.00013119
```

- ➔ We decided to further our selection by using the backward elimination method
- ➔ Using the backward elimination method, we were left with 14 recommended variables
- ➔ If we were to determine the regression formula by looking at **accommodates only**, it would be as follows,
  - ◆ **log_price = 1.200 + 0.0819 * accommodates**

Multiple Linear Regression - Summary

- The r-squared for our model is 0.5994.
  - This means that close to 60% of our selected variable points would fit on the regression line.
  - Our RMSE is 0.4444 which measures how well our model performed by measuring the difference between predicted values and actual values.
    - The closer the number is to 0, the better.
    - Performance of MLR = Slightly above Average.
- To improve, remove latitude and longitude since it might not affect the outcome of the log_price.
- However, to preserve our full data, we decided to leave those variables for now.

```
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)         1.200572410 35.063589633   0.034              0.97269
accommodates        0.081977701  0.005509317  14.880 < 0.0000000000000002 ***
bathrooms           0.123043962  0.022156263   5.553      0.0000000316196 ***
cleaning_feeTRUE   -0.090516797  0.022730271  -3.982      0.0000706401868 ***
host_response_rate -0.202889566  0.085820724  -2.364              0.01817 *
latitude            3.824310675  0.438371917   8.724 < 0.0000000000000002 ***
longitude           2.224965726  0.327595804   6.792      0.0000000000144 ***
number_of_reviews  -0.000462088  0.000220444  -2.096              0.03619 *
review_scores_rating 0.004012014  0.001308888  3.065              0.00220 **
property_type       0.002448051  0.001273690   1.922              0.05474 .
room_type          -0.611068006  0.023476376 -26.029 < 0.0000000000000002 ***
amenities           0.000001780  0.000000547   3.255              0.00115 **
bed_type            0.049221027  0.022263854   2.211              0.02716 *
instant_bookable   -0.066019242  0.020339278  -3.246              0.00119 **
neighbourhood       0.000131188  0.000054757   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: < 0.00000000000000022

                   ME      RMSE       MAE        MPE     MAPE
Test set -0.004012942 0.4444802 0.3298527 -0.9052584 6.880486
```

K Nearest Neighbors (knn)

- ➔ Classification approach
  - ◆ Created a new rental object instance and selected numerical predictors/attributes
    - Log_price, review_score_rating, beds, bathrooms, bedrooms, longitude, latitude, accommodates
  - ◆ The new object is assigned to the most common class among is neighbors measured by distance
    - Euclidean, Hamming, Correlation
  - ◆ Data Partitioning and Data Normalization
    - 60% training 40 % validation

KNN-New Object Creation & Model

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

```
install.packages('caret')
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")
library(FNN)

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

Predication & Neighbors

- We would have a cleaning fee based on the models prediction
- Our optimal K=Value is 9 based on the accuracy assessment

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

	k	accuracy
1	1	0.6990641
2	2	0.6040317
3	3	0.7444204
4	4	0.6904248
5	5	0.7667387
6	6	0.7336213
7	7	0.7631389
8	8	0.7530598
9	9	0.7717783
10	10	0.7696184

Naive Bayes

- ➔ In reference to price, most of our price selection lies in the below average rating
    - ◆ This would make sense since Boston is considered to be a college town.
    - ◆ Our apriori can confirm a probability of 49.56% that the majority of rentals fall in below average range

Categorical bins and A-Priori

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

A-priori probabilities:
Y

Student Budget	Below Average	Above Average	Pricey Dig
0.2331730769	0.4956730769	0.2706730769	0.0004807692

- ➔ To test our Naive Bayes model we created a fictional Apartment with the following variables:
    - ◆ Property_type - Apartment
    - ◆ Cancellation_policy - Flexible
    - ◆ Bed_type - Real bed
    - ◆ Cleaning_fee - True
- ➔ Output: Student Budget Price category

Prediction table from fictional case

	actual <fctr>	predicted <fctr>	Student.Budget <dbl>	Below.Average <dbl>	Above.Average <dbl>	Pricey.Dig <dbl>
16	Student Budget	Student Budget	0.9729938	0.01747418	0.009532015	9.937877e-11
38	Student Budget	Student Budget	0.9083229	0.05931906	0.032358040	3.373581e-10
86	Student Budget	Student Budget	0.9748305	0.01628573	0.008883729	9.261988e-11
148	Student Budget	Student Budget	0.9759222	0.01557935	0.008498403	8.860256e-11
183	Student Budget	Student Budget	0.9742467	0.01666350	0.009089799	9.476832e-11
199	Student Budget	Student Budget	0.6430391	0.23096919	0.125991719	1.313563e-09
203	Student Budget	Student Budget	0.9748305	0.01628573	0.008883729	9.261988e-11
215	Student Budget	Student Budget	0.6430391	0.23096919	0.125991719	1.313563e-09
290	Student Budget	Student Budget	0.7298860	0.17477546	0.095338521	9.939793e-10
292	Student Budget	Student Budget	0.6430391	0.23096919	0.125991719	1.313563e-09

Naive Bayes - Performance

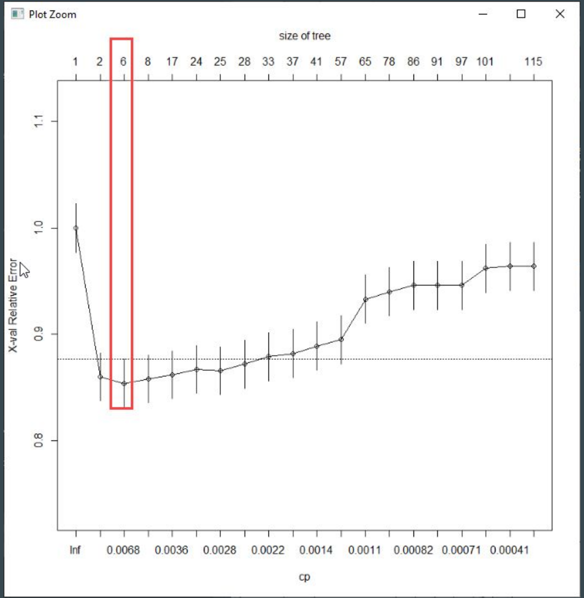
- A confusion matrix was created to the test the accuracy for both the training and validation sets
- The matrix on the right reflects the validation set
  - Validation Accuracy = 94.24%
  - Training Accuracy = 97.31%
  - This would make sense since,
    - Training slice = 60%
    - Validation slice = 40%

- The Naive Bayes model was successful and useful for this data

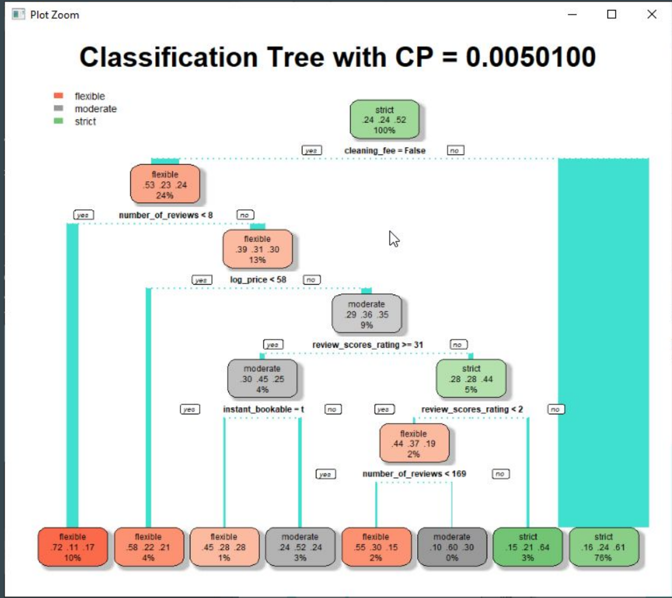
```
Confusion Matrix and Statistics

                Reference
Prediction       Student Budget Below Average Above Average Pricey Dig
  Student Budget            340             7             5          0
  Below Average               2           701            66          0
  Above Average               0             0           267          0
  Pricey Dig                  0             0             0          0

Overall Statistics

               Accuracy : 0.9424
                 95% CI : (0.9288, 0.954)
    No Information Rate : 0.5101
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9052

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Student Budget Class: Below Average Class: Above Average
Sensitivity                         0.9942               0.9901               0.7899
Specificity                         0.9885               0.9000               1.0000
Pos Pred Value                      0.9659               0.9116               1.0000
Neg Pred Value                      0.9981               0.9887               0.9367
Prevalence                          0.2464               0.5101               0.2435
Detection Rate                      0.2450               0.5050               0.1924
Detection Prevalence                0.2536               0.5540               0.1924
Balanced Accuracy                   0.9913               0.9451               0.8950
                     Class: Pricey Dig
Sensitivity                         NA
Specificity                          1
Pos Pred Value                      NA
Neg Pred Value                      NA
Prevalence                           0
Detection Rate                       0
Detection Prevalence                 0
Balanced Accuracy                   NA
```

Classification Tree - Overview

- ➔ Converted super_strict_30 and super_strict_60 to strict.
- ➔ Eliminated variables and ensured remaining variables fit format/structure necessary for Classification Tree
- ➔ Ran unpruned Classification Tree with CP = 0 to determine CP value with minimum cross-validation error (xerror)
- ➔ Input new xerror-minimizing CP value into model to build final Classification Tree



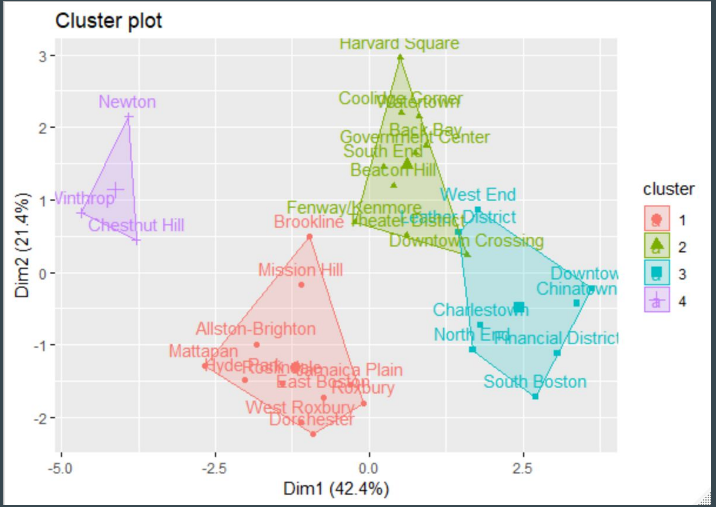
Best CP value from unpruned tree



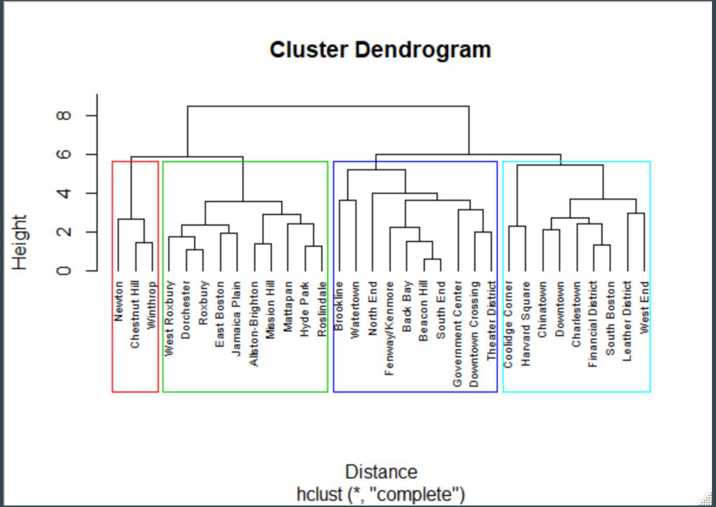
Final Classification Tree

Clustering - Overview

- ➔ Excluded ‘Cambridge’ and ‘Somerville’ as they had only 5 properties in total.
- ➔ Converted ‘cancellation_policy’ to a measurable variable with a scale of 1 (flexible) to 5 (super_strict_60)
- ➔ Created a new variable ‘price_per_person’ from ‘nightly_price’ and ‘accommodates’
- ➔ Optimum number of clusters were found to be at 4 (Elbow and Average Silhouette Methods)

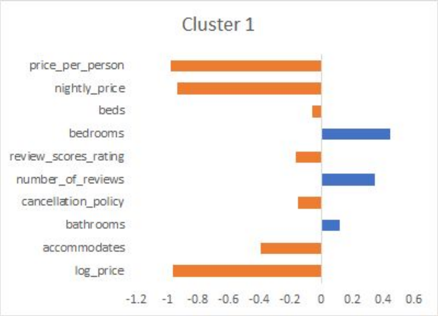


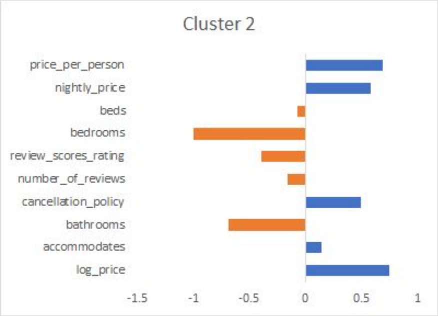
K-means

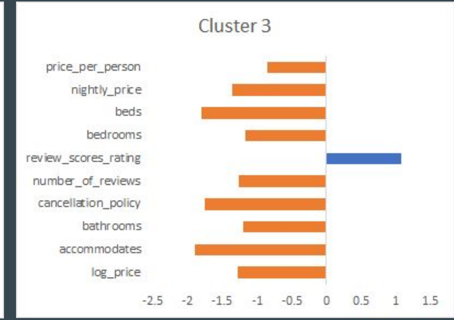


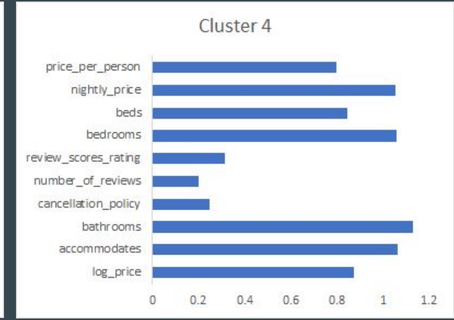
Hierarchical

Clustering - Descriptive Analysis









Cluster 1: You get what you pay for. Low price neighborhoods with lower avg. review ratings. However, you get more space for your buck.

Neighborhoods: *Newton, Chestnut Hill, Winthrop*

Cluster 2: You pay more for less space. The reviews are not so good either and the cancellation policy is stricter.

Neighborhoods: *West Roxbury, Dorchester, Roxbury, East Boston, Jamaica Plain, Allston-Brighton, Mission Hill, Mattapan, Hyde Park, Roslindale*

Cluster 3: Utopia neighborhoods for smaller groups of guests. Lenient cancellation policy.

Neighborhoods: *Brookline, Watertown, North End, Fenway/Kenmore, Back Bay, Beacon Hill, South End, Government Center, Downtown Crossing, Theater District*

Cluster 4: Expensive neighborhoods for the deep pockets. Well reviewed and roomier. However, be prepared to lose your deposit if your plans change.

Neighborhoods: *Coolidge Corner, Harvard Square, Chinatown, Downtown, Charlestown, Financial District, South Boston, Leather District, West End*

Conclusion

- For Naive Bayes, half of the visitors fall within the 'Below Average' price category. When you combine this with clustering, we can see what neighborhoods would fit this category. Based on our analysis and knowing that Boston is a college town, we can conclude that this audience are mostly college students.
- Seeing that our KNN model predicted we would have a cleaning fee we learned that most of our neighbors also are charging a similar fee based on the predictors we used and the accuracy of our model.
- Based on the Classification Tree determining Cancelation Policy, the most important factor was whether or not the Airbnb charged a Cleaning Fee. If the host charged a cleaning fee, the model predicted it would also have a Strict cancelation policy.
- Finding similar neighborhoods using ***Clustering***, predicting the cancelation policy of a property type using ***Classification Trees***, determining the category of a property using ***Naive Bayes*** and finally predicting the log price using ***Multiple Linear Regression***, each Data Mining method was trying to answer a specific real-world business question. Starting from Data Preparation and Exploration to running various analysis, the overall nature of this assignment was collaborative.