Austin Rodriquez 3/24/2023

Group 11 Part 2 of the assignment (Logistic Regression, knn, and Decision trees)

```
if(!require("vcd")){
  install.packages("vcd")
}
```

```
## Loading required package: vcd
```

```
## Loading required package: grid
```

```
if(!require("e1071")){
  install.packages("vcd")
}
```

```
## Loading required package: e1071
```

The data used is “Airbnb price prediction” by SteveZheng. The data can be accessed at: https://www.kaggle.com/datasets/stevezhenghp/airbnb-price-prediction?resource=download (https://www.kaggle.com/datasets/stevezhenghp/airbnb-price-prediction?resource=download)

```
df <- read.csv("pricePrediction.csv")
dim(df)
```

```
## [1] 74111    29
```

```
names(df)
```

```
##  [1] "id"                     "log_price"              "property_type"         
##  [4] "room_type"              "amenities"              "accommodates"          
##  [7] "bathrooms"              "bed_type"               "cancellation_policy"   
## [10] "cleaning_fee"           "city"                   "description"           
## [13] "first_review"           "host_has_profile_pic"   "host_identity_verified"
## [16] "host_response_rate"     "host_since"             "instant_bookable"      
## [19] "last_review"            "latitude"               "longitude"             
## [22] "name"                   "neighbourhood"          "number_of_reviews"     
## [25] "review_scores_rating"   "thumbnail_url"          "zipcode"               
## [28] "bedrooms"               "beds"
```

As one can tell, there’s are many columns. We don’t need most, so we will only keep the ones that are effective to work with and don’t have too many null values.

```
df <- df[,c(2, 3, 4, 6, 7, 8, 9, 10, 11, 15, 18, 24, 25, 28, 29)]
names(df)
```

```
##  [1] "log_price"              "property_type"          "room_type"             
##  [4] "accommodates"           "bathrooms"              "bed_type"              
##  [7] "cancellation_policy"    "cleaning_fee"           "city"                  
## [10] "host_identity_verified" "instant_bookable"       "number_of_reviews"     
## [13] "review_scores_rating"   "bedrooms"               "beds"
```

We are left with about half the columns. Next, the nulls will be removed since there are plenty of observations in the data frame.

```
df <- na.omit(df)
df <- subset(df, host_identity_verified != "")
dim(df)
```

```
## [1] 56989    15
```

The observations have dropped from 74,000 to 56,000.

Next, the column types will be investigated and changed if necessary.

```
str(df)
```

```
## 'data.frame':    56989 obs. of  15 variables:
##  $ log_price             : num  5.01 5.13 4.98 4.74 4.44 ...
##  $ property_type         : chr  "Apartment" "Apartment" "Apartment" "Apartment" ...
##  $ room_type             : chr  "Entire home/apt" "Entire home/apt" "Entire home/apt" "Entire home/apt" ...
##  $ accommodates          : int  3 7 5 2 2 3 2 2 2 2 ...
##  $ bathrooms             : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ bed_type              : chr  "Real Bed" "Real Bed" "Real Bed" "Real Bed" ...
##  $ cancellation_policy   : chr  "strict" "strict" "moderate" "moderate" ...
##  $ cleaning_fee          : chr  "True" "True" "True" "True" ...
##  $ city                  : chr  "NYC" "NYC" "NYC" "DC" ...
##  $ host_identity_verified: chr  "t" "f" "t" "t" ...
##  $ instant_bookable      : chr  "f" "t" "t" "t" ...
##  $ number_of_reviews     : int  2 6 10 4 3 15 9 159 2 82 ...
##  $ review_scores_rating  : num  100 93 92 40 100 97 93 99 90 93 ...
##  $ bedrooms              : num  1 3 1 0 1 1 1 1 1 1 ...
##  $ beds                  : num  1 3 3 1 1 1 1 1 1 1 ...
```

Columns to be converted to factor: property type, room type, bed types, cancellation policy, cleaning fee, city, host identity, bookable,

```
df$property_type <- factor(df$property_type)
df$room_type <- factor(df$room_type)
df$bed_type <- factor(df$bed_type)
df$cancellation_policy <- factor(df$cancellation_policy)
df$cleaning_fee <- factor(df$cleaning_fee)
df$city <- factor(df$city)
df$host_identity_verified <- factor(df$host_identity_verified)
df$instant_bookable <- factor(df$instant_bookable)

str(df)
```

```
## 'data.frame':    56989 obs. of  15 variables:
##  $ log_price             : num  5.01 5.13 4.98 4.74 4.44 ...
##  $ property_type         : Factor w/ 32 levels "Apartment","Bed & Breakfast",..: 1 1 1 1 1 11 17 17 1 ...
##  $ room_type             : Factor w/ 3 levels "Entire home/apt",..: 1 1 1 1 2 1 1 2 2 2 ...
##  $ accommodates          : int  3 7 5 2 2 3 2 2 2 2 ...
##  $ bathrooms             : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 2 2 3 2 2 2 2 3 ...
##  $ cleaning_fee          : Factor w/ 2 levels "False","True": 2 2 2 2 2 2 2 2 2 2 ...
##  $ city                  : Factor w/ 6 levels "Boston","Chicago",..: 5 5 5 3 6 4 4 6 4 5 ...
##  $ host_identity_verified: Factor w/ 2 levels "f","t": 2 1 2 2 2 1 2 1 1 2 ...
##  $ instant_bookable      : Factor w/ 2 levels "f","t": 1 2 2 2 2 2 1 1 2 1 ...
##  $ number_of_reviews     : int  2 6 10 4 3 15 9 159 2 82 ...
##  $ review_scores_rating  : num  100 93 92 40 100 97 93 99 90 93 ...
##  $ bedrooms              : num  1 3 1 0 1 1 1 1 1 1 ...
##  $ beds                  : num  1 3 3 1 1 1 1 1 1 1 ...
```

Since this data set is built to predict price and this assignment is to figure out a logistic regression, we will make a new column witch will determine what factors make up the 3rd quarterly of price.

```
summary(df)
```

```
##     log_price         property_type              room_type      accommodates
##  Min.   :0.000   Apartment   :37445   Entire home/apt:32836   Min.   : 1.000
##  1st Qu.:4.304   House       :12982   Private room   :22721   1st Qu.: 2.000
##  Median :4.700   Condominium: 1995   Shared room    : 1432   Median : 2.000
##  Mean   :4.750   Townhouse   : 1286                           Mean   : 3.224
##  3rd Qu.:5.165   Loft        : 1009                           3rd Qu.: 4.000
##  Max.   :7.600   Other       :  418                           Max.   :16.000
##                  (Other)     : 1854
##    bathrooms             bed_type           cancellation_policy cleaning_fee
##  Min.   :0.000   Airbed       :  342   flexible        :12893    False:11685
##  1st Qu.:1.000   Couch        :  158   moderate        :16391    True :45304
##  Median :1.000   Futon        :  599   strict          :27617
##  Mean   :1.227   Pull-out Sofa:  489   super_strict_30:   78
##  3rd Qu.:1.000   Real Bed     :55401   super_strict_60:   10
##  Max.   :8.000
##
##       city       host_identity_verified instant_bookable number_of_reviews
##  Boston : 2811   f:15448                f:41682          Min.   :  1.00
##  Chicago: 3204   t:41541                t:15307          1st Qu.:  3.00
##  DC     : 4083                                           Median : 11.00
##  LA     :17109                                           Mean   : 26.94
##  NYC    :24754                                           3rd Qu.: 33.00
##  SF     : 5028                                           Max.   :605.00
##
##  review_scores_rating    bedroom          beds
##  Min.   : 20.00       Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 92.00       1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 96.00       Median : 1.000   Median : 1.000
##  Mean   : 94.08       Mean   : 1.262   Mean   : 1.741
##  3rd Qu.:100.00       3rd Qu.: 1.000   3rd Qu.: 2.000
##  Max.   :100.00       Max.   :10.000   Max.   :18.000
##
```

The 3rd quarterly of price is 5.165. The log price column will be converted to a factor with 1 representing the price is of the 3rd quarter and 0 if not.It will also be renamed to "expensive".

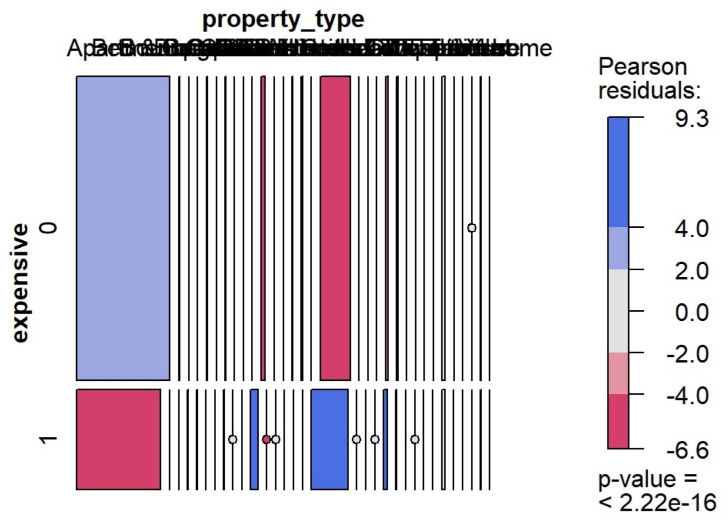
```
df$log_price <- as.factor(ifelse (df$log_price >= 5.165, 1, 0))
names(df)[names(df) == 'log_price'] <- 'expensive'
```

a. Next, the data will be split into 80% train and 20% test. This will leave train with 45,000 entries and test with 11,000 entries.

```
set.seed(123)
i <- sample(1:nrow(df), nrow(df) * .80, replace = FALSE)
train <- df[i,]
test <- df[-i,]
```

b. Since determining if the room is expensive is the goal, each other column will be compared to determine if there's an influence.

```
mosaic(table(train[,c(1,2)]), shade = TRUE, legend = TRUE)
```

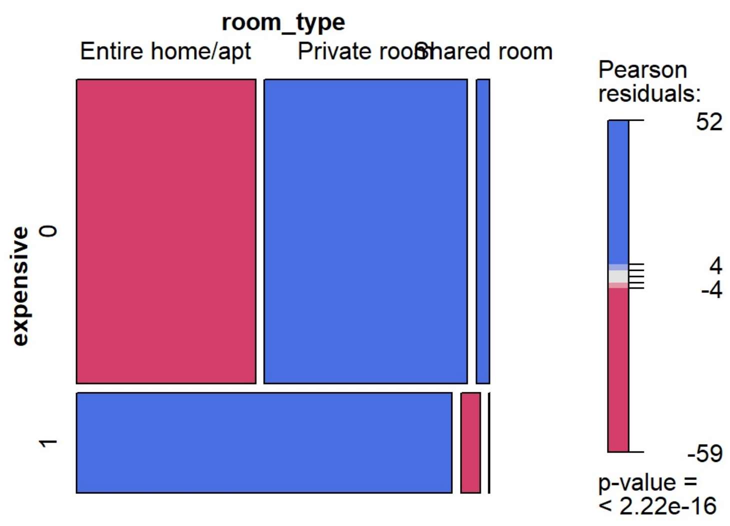


```
levels(train$property_type)
```

```
##  [1] "Apartment"          "Bed & Breakfast"    "Boat"
##  [4] "Boutique hotel"     "Bungalow"           "Cabin"
##  [7] "Camper/RV"          "Castle"             "Cave"
## [10] "Chalet"             "Condominium"        "Dorm"
## [13] "Earth House"        "Guest suite"        "Guesthouse"
## [16] "Hostel"             "House"              "Hut"
## [19] "In-law"             "Island"             "Loft"
## [22] "Other"              "Serviced apartment" "Tent"
## [25] "Timeshare"          "Tipi"               "Townhouse"
## [28] "Train"              "Treehouse"          "Vacation home"
## [31] "Villa"              "Yurt"
```

It doesn't seem like property type could identify the expensive apartments much. Every type has roughly the same percentage whether is cheap or expensive.

```
mosaic(table(train[,c(1,3)]), shade = TRUE, legend = TRUE)
```

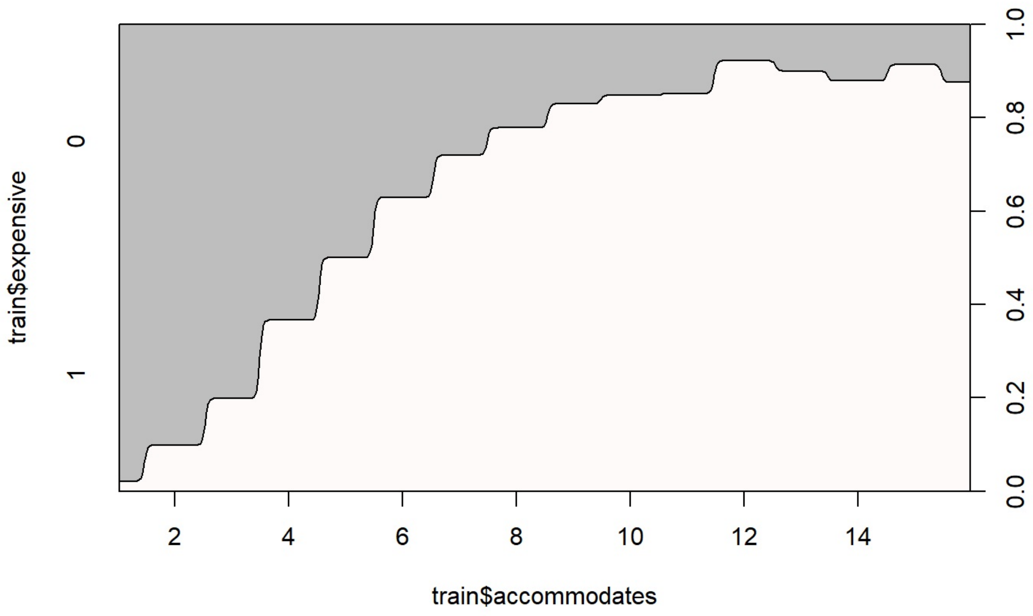


```
levels(train$room_type)
```

```
## [1] "Entire home/apt" "Private room"    "Shared room"
```

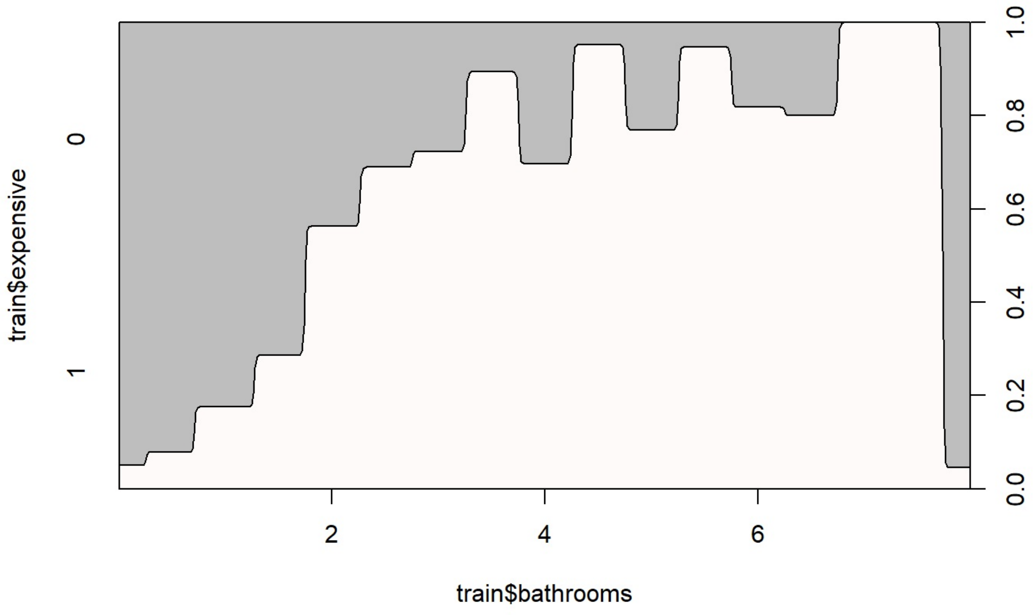
Getting the entire home indicates it is very likely to be of the most expensive.It's very unlikely a shared room will be expensive.

```
cdplot(train$accommodates, train$expensive, col=c("snow", "gray"))
```



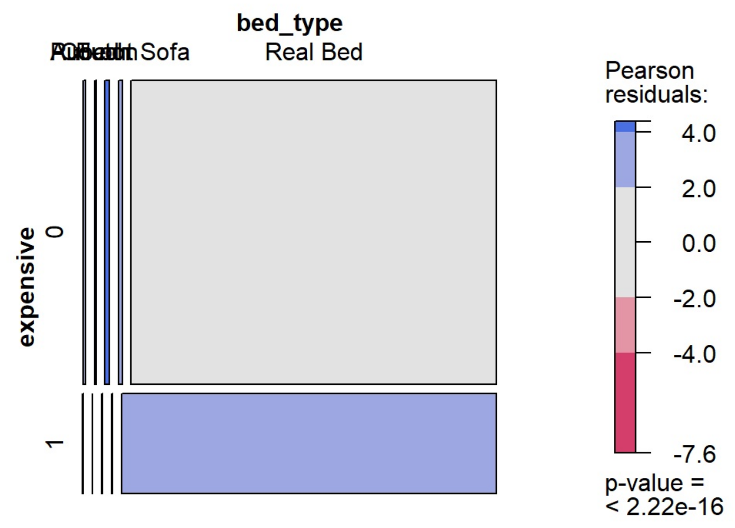
Naturally, as the accommodations increase, it is more likely to be expensive.

```
cdplot(train$bathrooms, train$expensive, col=c("snow", "gray"))
```



The data does get a bit odd, but it shows that having more bathrooms means the room is going to be expensive.

```
mosaic(table(train[,c(1,6)]), shade = TRUE, legend = TRUE)
```

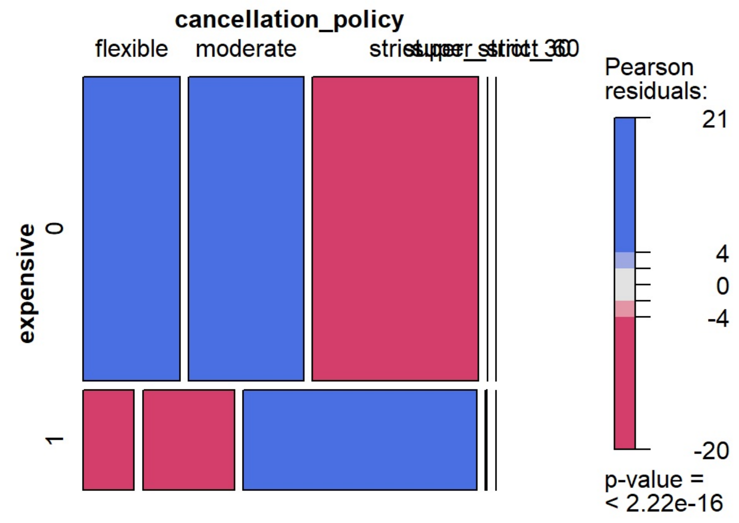


```
levels(train$bed_type)
```

```
## [1] "Airbed"        "Couch"         "Futon"         "Pull-out Sofa"
## [5] "Real Bed"
```

Bed type doesn't affect the price much. They mostly all just have real beds.

```
mosaic(table(train[,c(1,7)]), shade = TRUE, legend = TRUE)
```

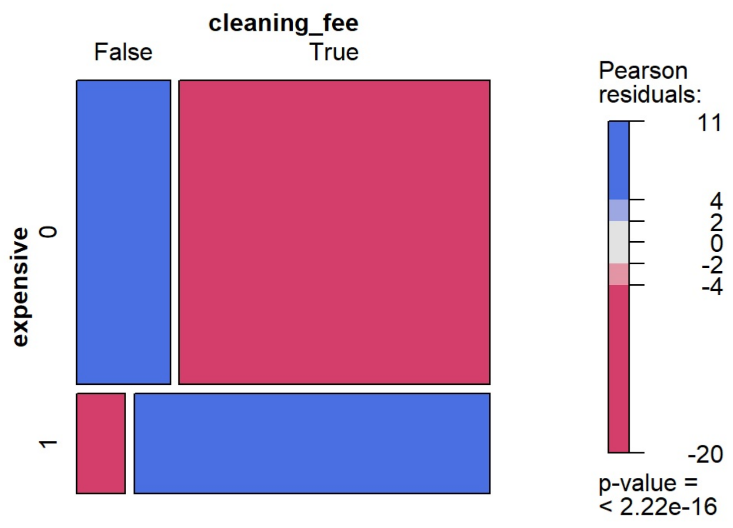


```
levels(train$cancellation_policy)
```

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

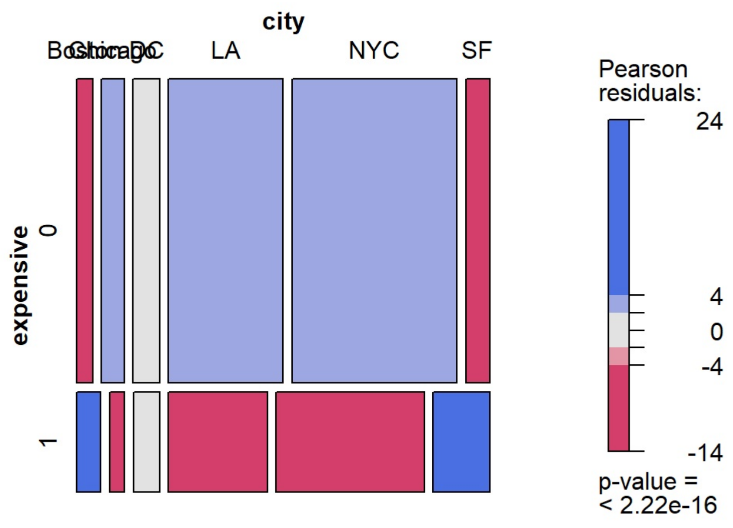
The more expensive apartments have stricter cancellation policies.

```
mosaic(table(train[,c(1,8)]), shade = TRUE, legend = TRUE)
```



It's not by much, but the expensive apartments are more likely to have a cleaning fee.

```
mosaic(table(train[,c(1,9)]), shade = TRUE, legend = TRUE)
```

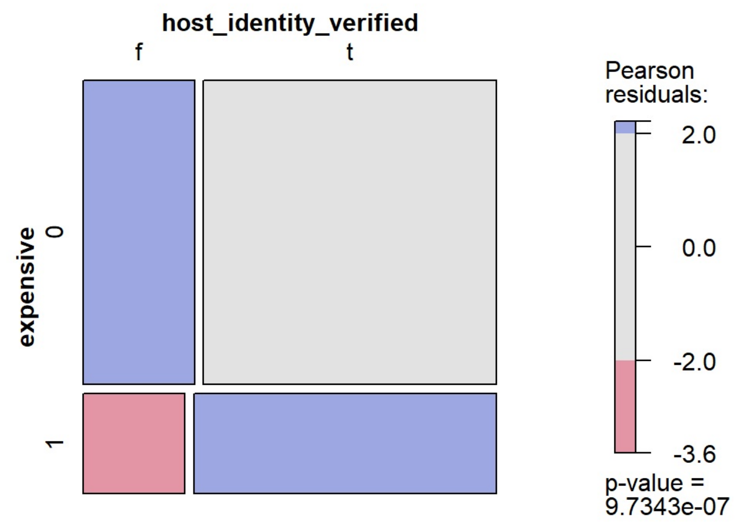


```
levels(train$city)
```

```
## [1] "Boston"  "Chicago" "DC"      "LA"      "NYC"     "SF"
```

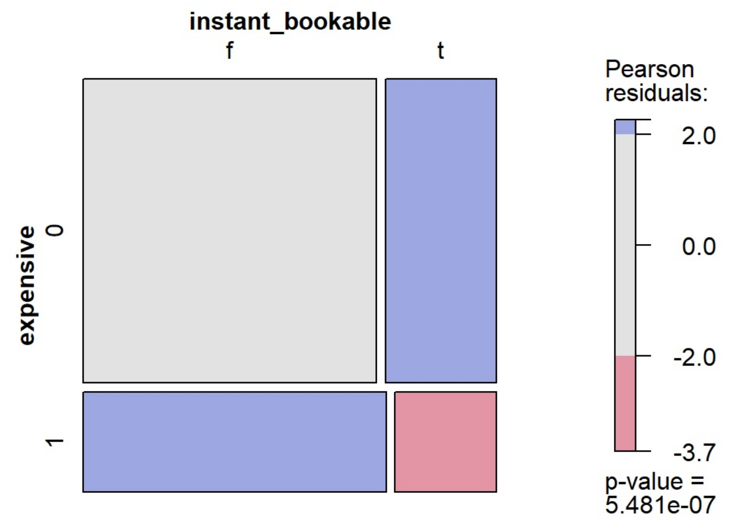
The city doesn't affect the price much.

```
mosaic(table(train[,c(1,10)]), shade = TRUE, legend = TRUE)
```



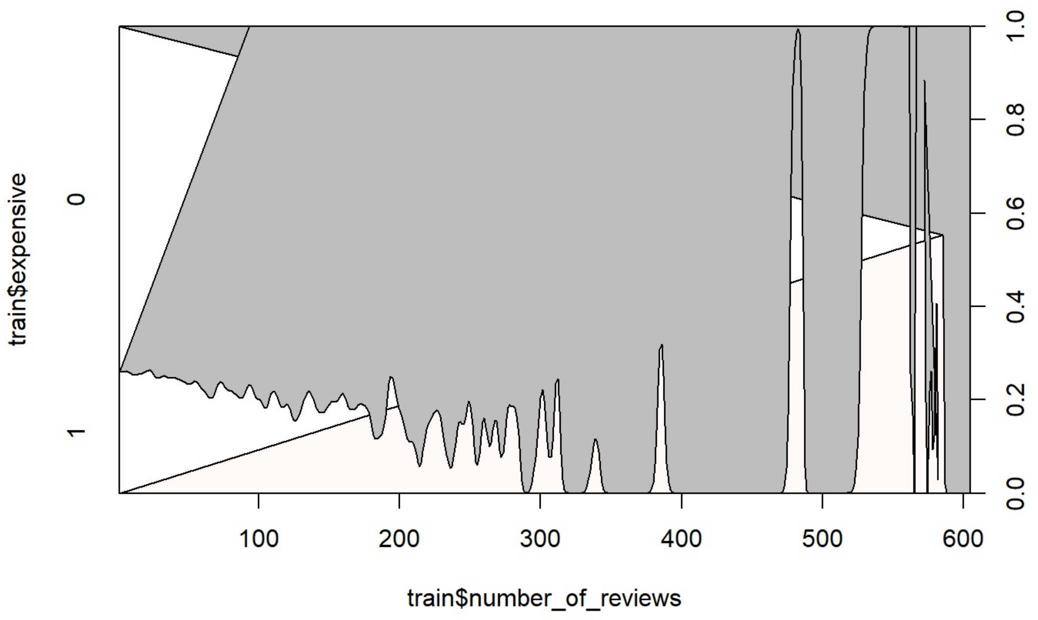
Whether or not the host has been verified doesn't affect the price much.

```
mosaic(table(train[,c(1,11)]), shade = TRUE, legend = TRUE)
```



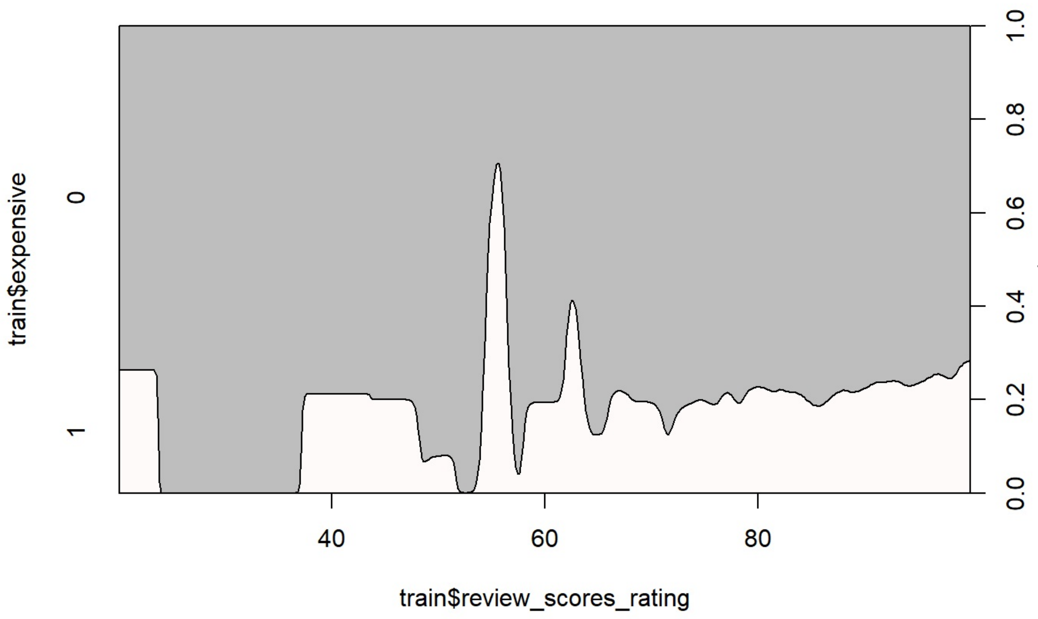
The instant bookablity also doesn't affect much.

```
cdplot(train$number_of_reviews, train$expensive, col=c("snow", "gray"))
```



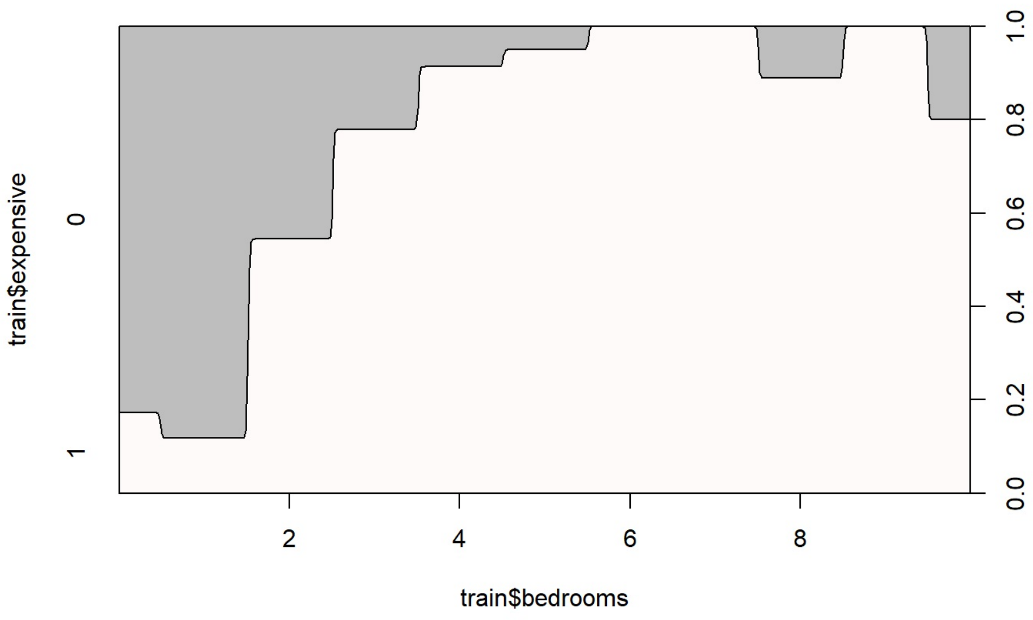
I have no clue what went wrong here.

```
cdplot(train$review_scores_rating, train$expensive, col=c("snow", "gray"))
```



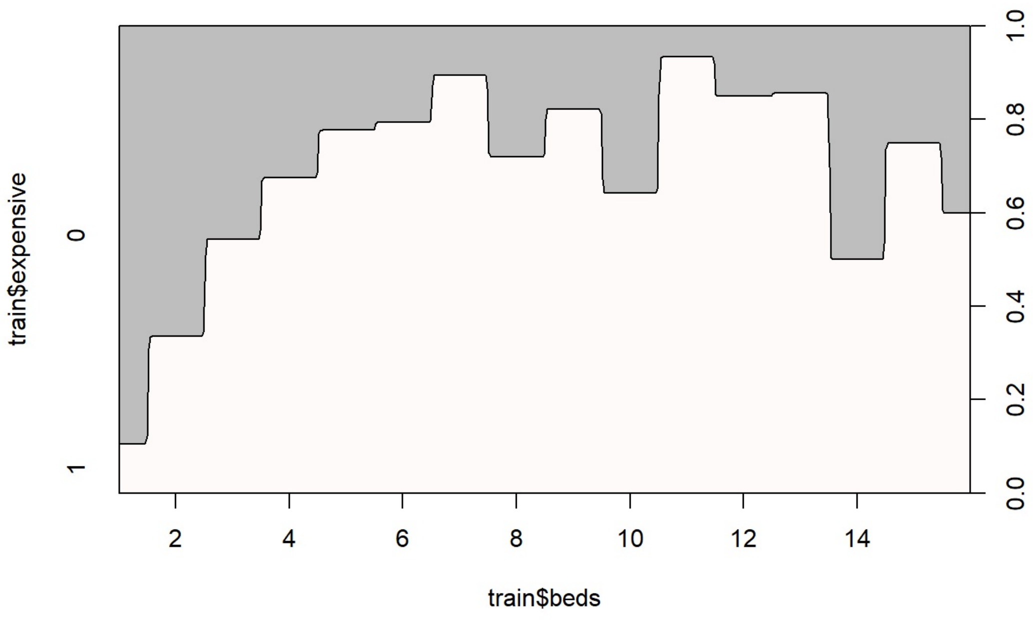
The chances of pretty much any review just claim there’s a 20% chance of it being an expensive place. It’s unlikely price has much to do with the review in this case.

```
cdplot(train$bedrooms, train$expensive, col=c("snow", "gray"))
```



More bedrooms indicate a more expensive place.

```
cdplot(train$beds, train$expensive, col=c("snow", "gray"))
```



The graph shows some odd fluctuation, but it seems that more beds does indicate a more expensive place. However, too many beds can represent overcrowding, depending on the number of rooms and the size of the rooms.

The worthwhile columns to consider are: room_type, accommodations, bathrooms, cancellation policy, cleaning fee, bedrooms.

c. A logistic regression model will be built with these columns as factors.

```
glm1 <- glm(expensive~room_type+accommodates+bathrooms+cancellation_policy+cleaning_fee+bedrooms, data = train, f
amily = binomial)
summary(glm1)
```

```
##
## Call:
## glm(formula = expensive ~ room_type + accommodates + bathrooms +
##     cancellation_policy + cleaning_fee + bedrooms, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9476  -0.6889  -0.2369  -0.0784   3.4275
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -2.957209   0.055948 -52.857  < 2e-16 ***
## room_typePrivate room             -2.544323   0.049097 -51.823  < 2e-16 ***
## room_typeShared room              -4.305442   0.288108 -14.944  < 2e-16 ***
## accommodates                       0.159624   0.009896  16.131  < 2e-16 ***
## bathrooms                          0.769518   0.031386  24.518  < 2e-16 ***
## cancellation_policymoderate        0.088660   0.042781   2.072  0.03822 *
## cancellation_policystrict          0.390730   0.039227   9.961  < 2e-16 ***
## cancellation_policysuper_strict_30  1.936622   0.308668   6.274 3.52e-10 ***
## cancellation_policysuper_strict_60  1.568734   1.198154   1.309  0.19044
## cleaning_feeTrue                  -0.124580   0.040047  -3.111  0.00187 **
## bedrooms                           0.586938   0.022940  25.586  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 51108  on 45590  degrees of freedom
## Residual deviance: 33661  on 45580  degrees of freedom
## AIC: 33683
##
## Number of Fisher Scoring iterations: 7
```

```
#expensive~room_type+accomodates+bathrooms+cancellation_policy+cleaning_fee+bedrooms
```

Its good that the residual deviance is much lower than the null deviance. Most of the factors were helpful, but some like policy moderate and policy strict_60 didn’t get many stars, meaning they weren’t very useful.

Most earlier assumptions were correct. For example, having anything but the whole house as the room type heavily indicates it’s less likely to be expensive. A shared room all but guarantees it’s not expensive. The strict cancellation policies also indicate the place is expensive.

Now the accuracy of the model will be examined.

```
probs <- predict(glm1, newdata=test, type="response")
pred <- ifelse(probs>0.5, 2, 1)
acc1 <- mean(pred==as.integer(test$expensive))
print(paste("Logistic regression accuracy = ", acc1))
```

```
## [1] "Logistic regression accuracy =  0.835848394455168"
```

```
table(pred, as.integer(test$expensive))
```

```
##
## pred    1    2
##    1 8072 1382
##    2  489 1455
```

The model had a pretty good 83.5% accuracy rate.

Next, a knn regression will be made.

The train and test will have to be split up into the data and labels.

We will attempt with the 3 numeric columns that were used in the logistic regression. The qualitative columns are incompatible with knn. Also, we can only use about 5% of the original data. Using any more than than constantly results in the “too many ties error” no matter what value of k is used.

```
train.knn <- train[1:2279,c(4, 5, 14)]
train.knnLabels <- train[1:2279,c(1)]

test.knn <- test[1:569,c(4, 5, 14)]
test.knnLabels <- test[1:569,c(1)]
```

Next, the knn model will actually be built.

```
library(class)
expensive_pred <- knn(train=train.knn, test=test.knn, cl=train.knnLabels, k = 5)
```

The accuracy will be tested next.

```
results1 <- expensive_pred == test.knnLabels
acc2 <- length(which(results1==TRUE)) / length((results1))
table(results1, expensive_pred)
```

```
##         expensive_pred
## results1   0   1
##    FALSE  57  35
##    TRUE  389  88
```

```
acc2
```

```
## [1] 0.8383128
```

Despite only using 5% of the data, the knn was able to get an accuracy of 83.8%. The model has a bad habit of making false positives.

In an attempt to improve results, the columns will be scaled and the model rebuilt and tested.

```
train.knnScaled <- train.knn[]
means <- sapply(train.knnScaled, mean)
stdvs <- sapply(train.knnScaled, sd)
train.knnScaled <- scale(train.knnScaled, center=means, scale=stdvs)
test.knnScaled <- scale(test.knn, center=means, scale=stdvs)
```

```
expensive_pred2 <- knn(train=train.knnScaled, test=test.knnScaled, cl=train.knnLabels, k = 3)
```

```
results2 <- expensive_pred2 == test.knnLabels
acc3 <- length(which(results2==TRUE)) / length((results2))
table(results2, expensive_pred2)
```

```
##         expensive_pred2
## results2   0   1
##    FALSE  56  31
##    TRUE  393  89
```

```
acc3
```

```
## [1] 0.8471002
```

There was a slight bump up in accuracy to 84.7% using scaled data. The false positive rate was not affected much.

Moving on, decision trees will be tried.

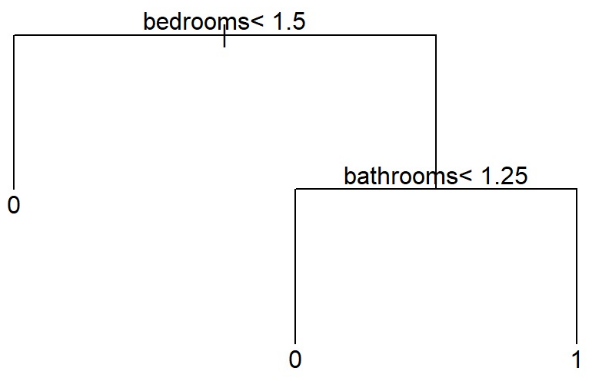
```
library(rpart)
tree_df <- rpart(expensive~room_type+accommodates+bathrooms+cancellation_policy+cleaning_fee+bedrooms, data=train
, method = "class")

tree_df
```

```
## n= 45591
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 45591 11322 0 (0.7516615 0.2483385)
##   2) bedrooms< 1.5 34634  4319 0 (0.8752960 0.1247040) *
##   3) bedrooms>=1.5 10957  3954 1 (0.3608652 0.6391348)
##     6) bathrooms< 1.25 5139  2415 0 (0.5300642 0.4699358) *
##     7) bathrooms>=1.25 5818  1230 1 (0.2114129 0.7885871) *
```

The tree was able to be made with the same six columns as were used in the logistic regression.

Next, the tree will be examined.

```
plot(tree_df, uniform=TRUE, margin=0.2)
text(tree_df)
```



The tree has interestingly enough only used the bedrooms and bathrooms as splits, despite have four other columns to use.

Next, the accuracy of this simple model will be tested.

```
pred4 <-predict(tree_df, newdata = test, type = "class")
table(pred4, test$expensive)
```

```
##
## pred4    0    1
##     0 8253 1681
##     1  308 1156
```

```
mean(pred4==test$expensive)
```

```
## [1] 0.8254957
```

Despite only using those two factors, it attained a 82.3% accuracy rate. It's not too far behind the logistic regression and knn models at all.

d. All three models determined that the largest factor in determining if a place is in the 3rd quarterly of price is it's size. The bigger the place, the more it'll cost.

That seems obvious now, but initial guesses were that things such as review score would be a major influence too. However, thinking about it, people often consider how much they paid when writing a review, so good or bad reviews aren't separated by price.

The single most effective factor was the number of rooms. That makes sense, considering that is the clearest indicator of size. It's worth noting that the maximum number of accommodations wasn't as useful in this case. It's possible that many being are just being stuffed into a smaller and less expensive place.

The second most influential factor was the number of bathrooms. This more than likely helped to determine if the issue of cramming was happening. If the accommodations were high and the bathrooms were plentiful as well, then it mostly likely means that the guests aren't being crammed in. Also, people just prefer having their own bathrooms, so that's expected to be a more high end luxury.

And so, this is what led to the fairly consistent and even accuracy of all three models.

Logistic Regression The logistic regression worked well since the data has a mostly liner trend. logistic regression works well with linear data. As size increases, price increases. However, the liner trend isn't perfect and this led to the decrease in accuracy since this model doesn't catch that complexity well.

Knn The knn did work well, but only after heavily altering the data. Only the numeric columns could be used, and the train data had to be reduced to only 5% of what it originally was in order to work. For that, I must say the end results aren't nearly as assured as compared to the other two models that used thousands of more entries to test and train.

However, it makes sense that the resulting accuracy was fairly high still. As discussed, the more room and bathrooms indicate a more expensive place. Given any value, the nearest neighbors are a good indicator of what kind of place you're getting.

Decision trees The results of this one were the most surprising and I'd consider it the winner. Even though it was a little less accurate, consider the fact it was able to train more heavily as compared the the knn model. It also figured out it could do away with columns such as cancellation policy that the logistic model tried using. It used only 2 out of the 6, but was only about 1% less accurate. It was by far the most interpretable.

The major weakness of trees being sensitive to noise was mitigated by the thousands of entries available. It was able to easily figure out that past 1.5 rooms and bathroom means the place is mostly likely in the upper echelon of cost.