Predicting Air BnB Prices

Andrew Cohen

1/10/2020

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
library(readr)
library(ggplot2)
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```r
library(rsample)
```

```
## Loading required package: tidyr
```

```r
df_listings <- read_csv("/Users/cohean/Desktop/DataSciChallenge/listings.csv",
                        col_types = cols(host_id = col_character(),
                                         id = col_character()))
# EDA
# library(rpivotTable)
# rpivotTable(df_listings)
```

```r
summary(df_listings)
```

```
##       id                 name              host_id         
##  Length:48864       Length:48864       Length:48864      
##  Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character  
##                                                          
##                                                          
##                                                          
##                                                          
##   host_name         neighbourhood_group neighbourhood         latitude    
##  Length:48864       Length:48864        Length:48864       Min.   :40.50  
##  Class :character   Class :character    Class :character   1st Qu.:40.69  
##  Mode  :character   Mode  :character    Mode  :character   Median :40.72  
##                                                            Mean   :40.73  
##                                                            3rd Qu.:40.76  
##                                                            Max.   :40.91  
##                                                                           
##    longitude       room_type         minimum_nights    
##  Min.   :-74.24   Length:48864       Min.   :   1.000  
##  1st Qu.:-73.98   Class :character   1st Qu.:   1.000  
##  Median :-73.96   Mode  :character   Median :   2.000  
##  Mean   :-73.95                      Mean   :   7.093  
##  3rd Qu.:-73.94                      3rd Qu.:   5.000  
##  Max.   :-73.71                      Max.   :1250.000  
##                                                        
##  calculated_host_listings_count availability_365 number_of_reviews
##  Min.   :  1.000                Min.   :  0.0    Min.   :  0.00   
##  1st Qu.:  1.000                1st Qu.:  0.0    1st Qu.:  1.00   
##  Median :  1.000                Median : 41.0    Median :  5.00   
##  Mean   :  7.438                Mean   :112.5    Mean   : 23.39   
##  3rd Qu.:  2.000                3rd Qu.:232.0    3rd Qu.: 24.00   
##  Max.   :343.000                Max.   :365.0    Max.   :639.00   
##                                                                   
##  reviews_per_month     price        
##  Min.   : 0.010    Min.   :    0.0  
##  1st Qu.: 0.190    1st Qu.:   69.0  
##  Median : 0.710    Median :  105.0  
##  Mean   : 1.366    Mean   :  151.5  
##  3rd Qu.: 2.000    3rd Qu.:  175.0  
##  Max.   :66.610    Max.   :10000.0  
##  NA's   :10131
```

```r
# remove nuisance columns
# additional analysis thoughts, potentially can include the "name" column using nlp techniques
df_listings <- subset(df_listings, select = -c(id, name, host_id, host_name, neighbourhood))

# check categorical for errors vars before encoding
df_listings %>% count(neighbourhood_group, sort = TRUE)
```

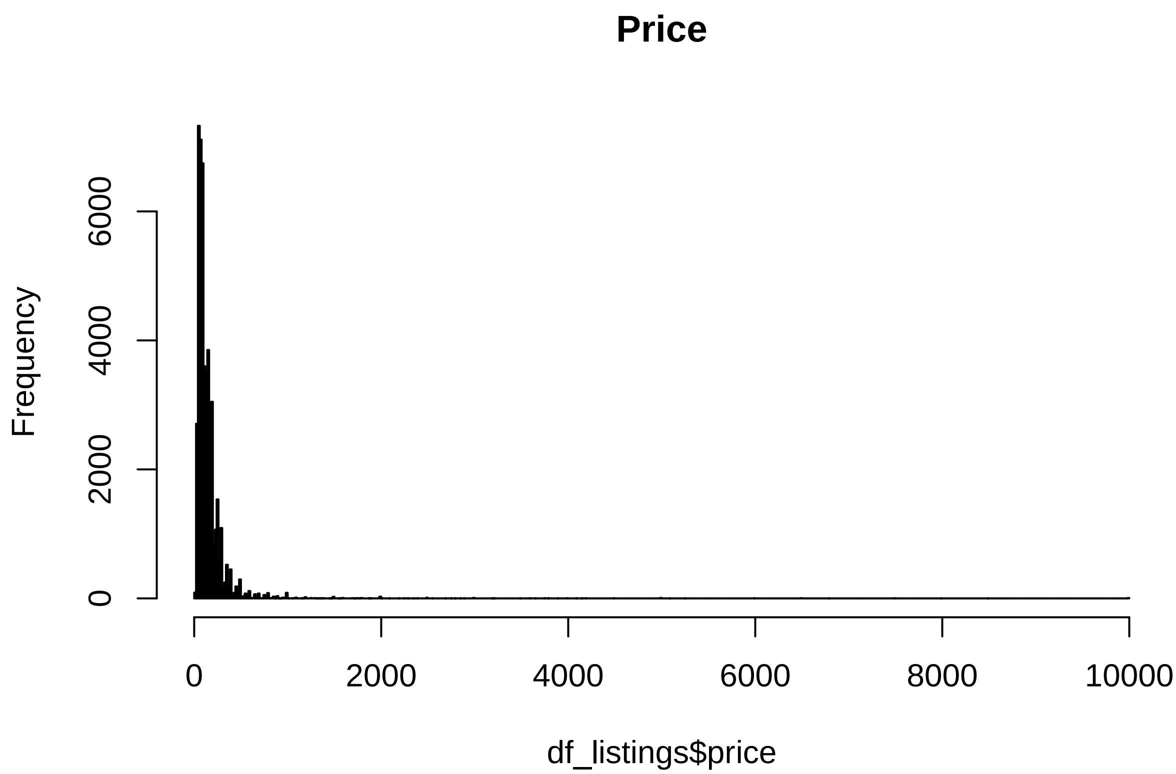
```
## # A tibble: 5 x 2
##   neighbourhood_group     n
##   <chr>               <int>
## 1 Manhattan           21456
```

```
## 2 Brooklyn            20114
## 3 Queens               5811
## 4 Bronx                1105
## 5 Staten Island         378
```

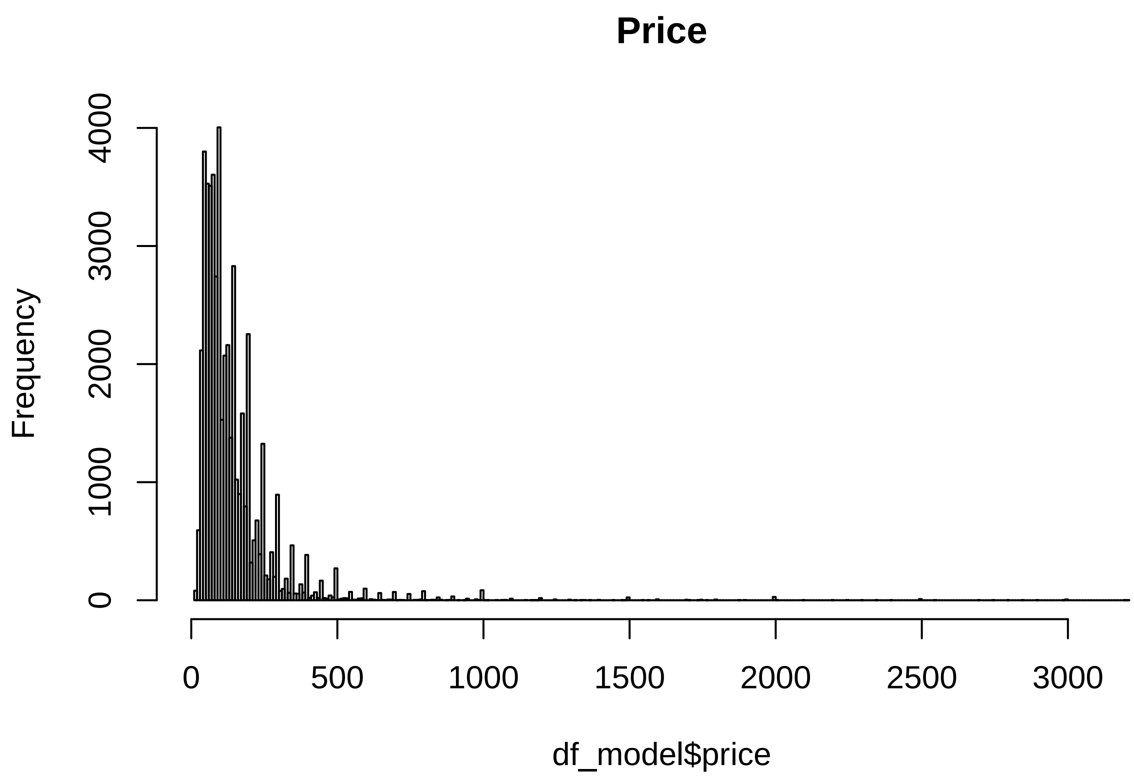
```r
df_listings %>% count(room_type, sort = TRUE)
```

```
## # A tibble: 3 x 2
##   room_type           n
##   <chr>           <int>
## 1 Entire home/apt 25296
## 2 Private room    22397
## 3 Shared room      1171
```

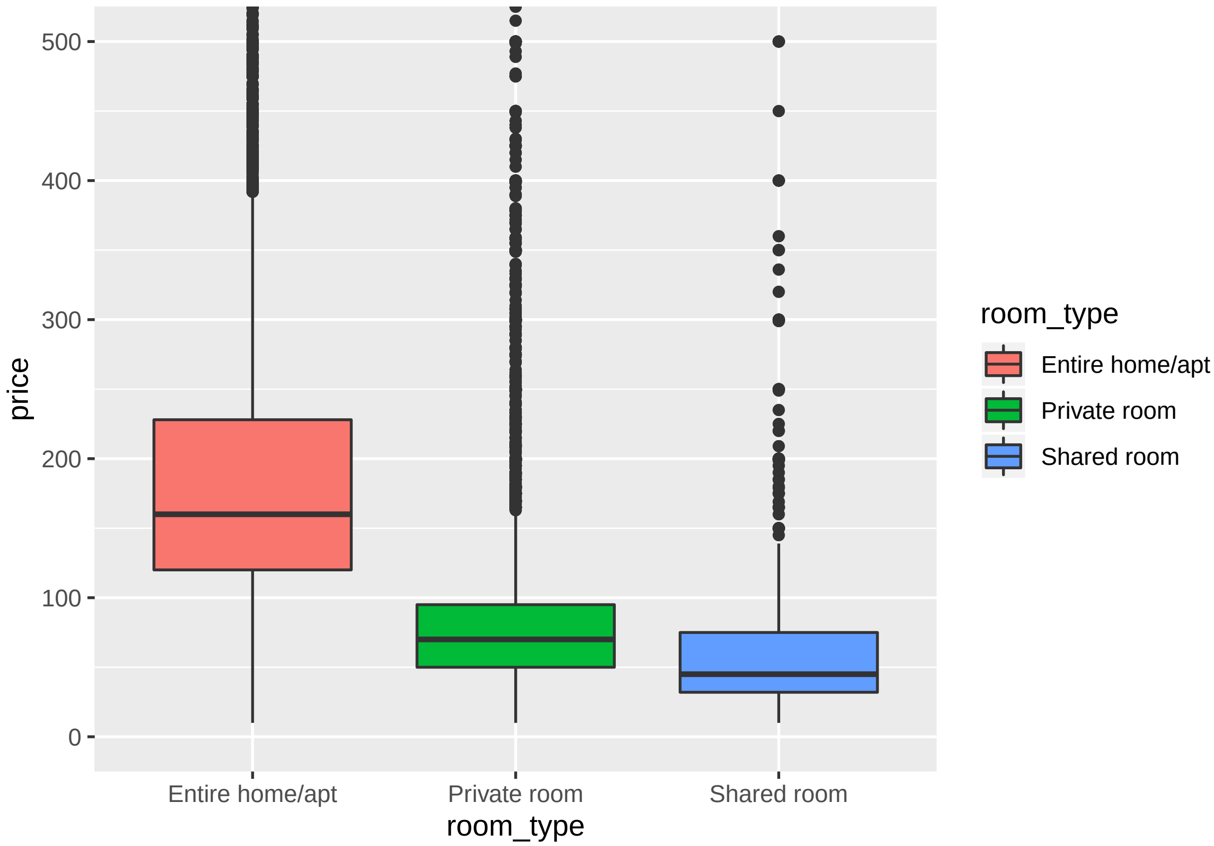
```r
#df_listings %>% count(neighbourhood, sort = TRUE)


# lapply(df_listings,class)
hist(df_listings$price, breaks = 500, main = "Price")
```



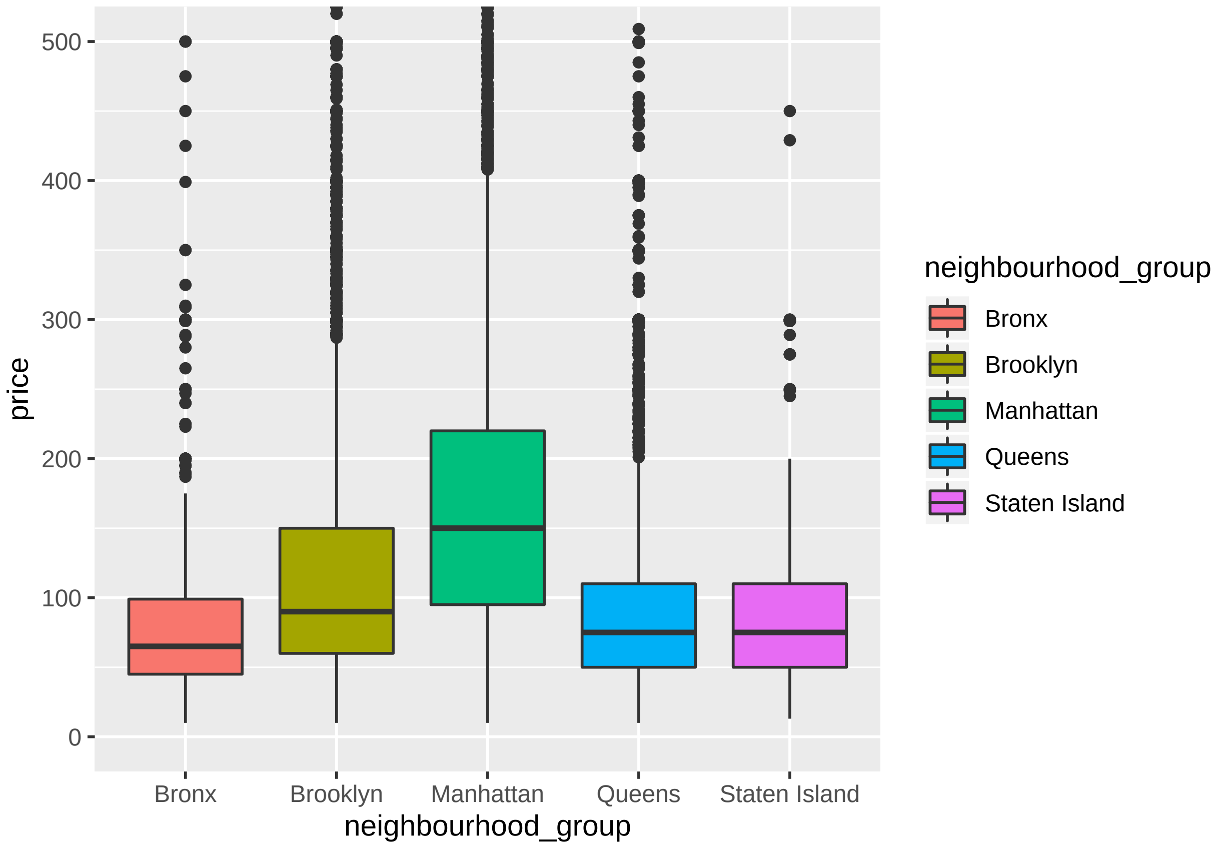

```r
# boxplot(df_listings$price)

#remove outliers
df_model <- df_listings[df_listings$price<3500,]
# remove zeros
df_model <- df_model[df_model$price != 0,]
hist(df_model$price, breaks = 250, main = "Price")
```



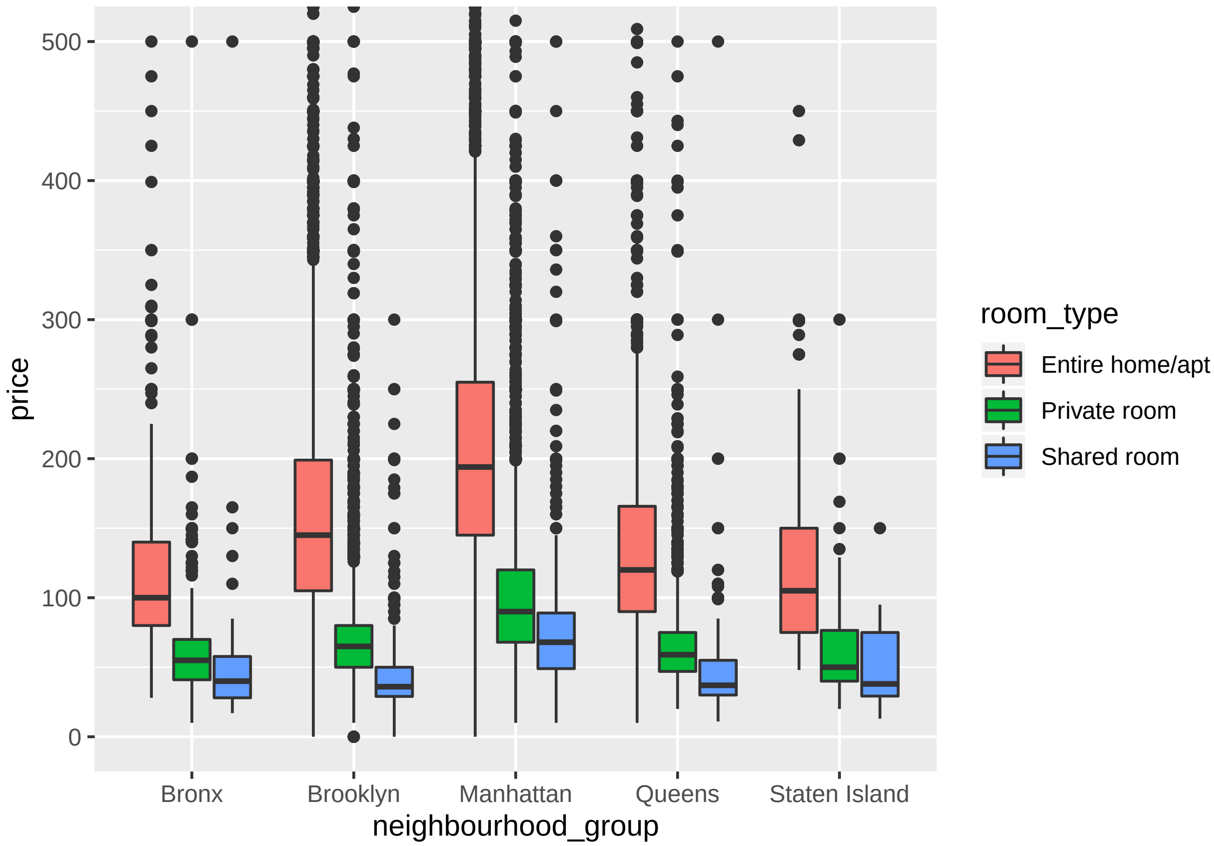

```
# boxplot room type
ggplot(df_model, aes(x=room_type, y=price, fill=room_type)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 500)) #zoom into center of data
```




```
# boxplot boro
ggplot(df_model, aes(x=neighbourhood_group, y=price, fill=neighbourhood_group)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 500)) #zoom into center of data
```




```
# grouped boxplot
ggplot(df_listings, aes(x=neighbourhood_group, y=price, fill=room_type)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 500)) #zoom into center of data
```




```
numericV <- which(sapply(df_model, is.numeric))
cor(df_model$price,df_model[,numericV])
```

```
##        latitude  longitude minimum_nights calculated_host_listings_count
## [1,] 0.04812225 -0.2158065     0.04180992                     0.09620459
##      availability_365 number_of_reviews reviews_per_month price
## [1,]        0.1047912       -0.05645767                NA     1
```

```
#not much correlation among numeric vars
```

```
# impute mean for missing to keep from loosing data
df_imp <- transform(df_model,
                    reviews_per_month = ifelse(is.na(reviews_per_month),
                                               mean(reviews_per_month, na.rm=TRUE), reviews_per_month))

# split data for models
df_split <- initial_split(df_imp, prop = .7)
df_train <- training(df_split)
df_test  <- testing(df_split)

# one-hot
df_dummy_train <- dummy.data.frame(df_train, names = c("neighbourhood_group","room_type") , sep = ".")
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =
## FALSE): non-list contrasts argument ignored

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =
## FALSE): non-list contrasts argument ignored
```

```r
df_dummy_test <- dummy.data.frame(df_test, names = c("neighbourhood_group","room_type") , sep = ".")
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =
## FALSE): non-list contrasts argument ignored

## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts =
## FALSE): non-list contrasts argument ignored
```

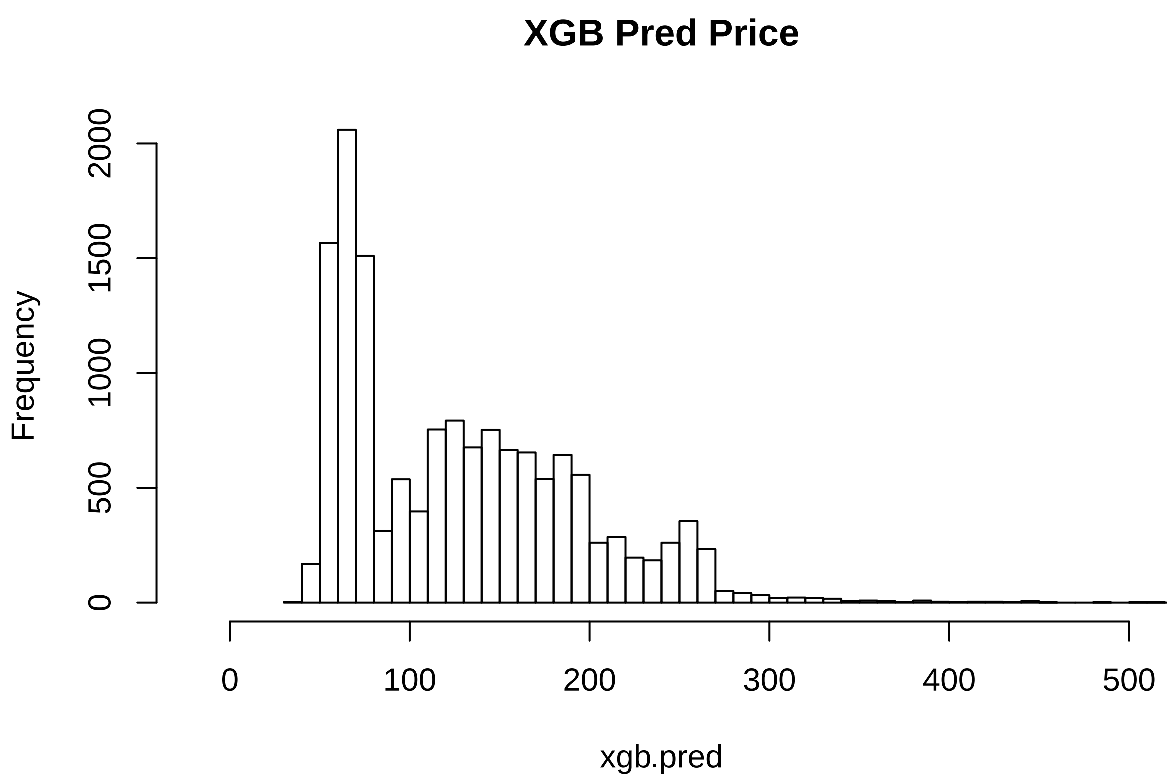
```r
# seperate X and Y matrices
df_train_x <- subset(df_dummy_train, select = -c(price))
df_train_y <- subset(df_dummy_train, select = c(price))
df_test_x <- subset(df_dummy_test, select = -c(price))
df_test_y <- subset(df_dummy_test, select = c(price))

# using xgboost is one of the best places to start for a predictive model because it usually fits very
xgb.train <- xgb.DMatrix(data = as.matrix(df_train_x), label=as.matrix(df_train_y))
xgb.test <- xgb.DMatrix(data = as.matrix(df_test_x), label=as.matrix(df_test_y))

# parameters based on some light tuning using regression performance metrics like rmse
params <- list(
  booster = "dart",
  #objective = "reg:gamma",
  max.depth = 5,
  eta = 0.007,
  #subsample = 0.60,
  eval_metric = "rmse"
  # ,eval_metric = "mae"
  )

xgb.fit<-xgb.train(
        data = xgb.train,
        params = params,
        nrounds = 300, # cut off based on rmse
        #watchlist = list(test=xgb.test,train=xgb.train),
        #verbose = 1
        )

# performance
xgb.fit
```

```
## ##### xgb.Booster
## raw: 681 Kb
## call:
##   xgb.train(params = params, data = xgb.train, nrounds = 300)
## params (as set within xgb.train):
##   booster = "dart", max_depth = "5", eta = "0.007", eval_metric = "rmse", silent = "1"
## xgb.attributes:
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
## # of features: 15
## niter: 300
## nfeatures : 15
```

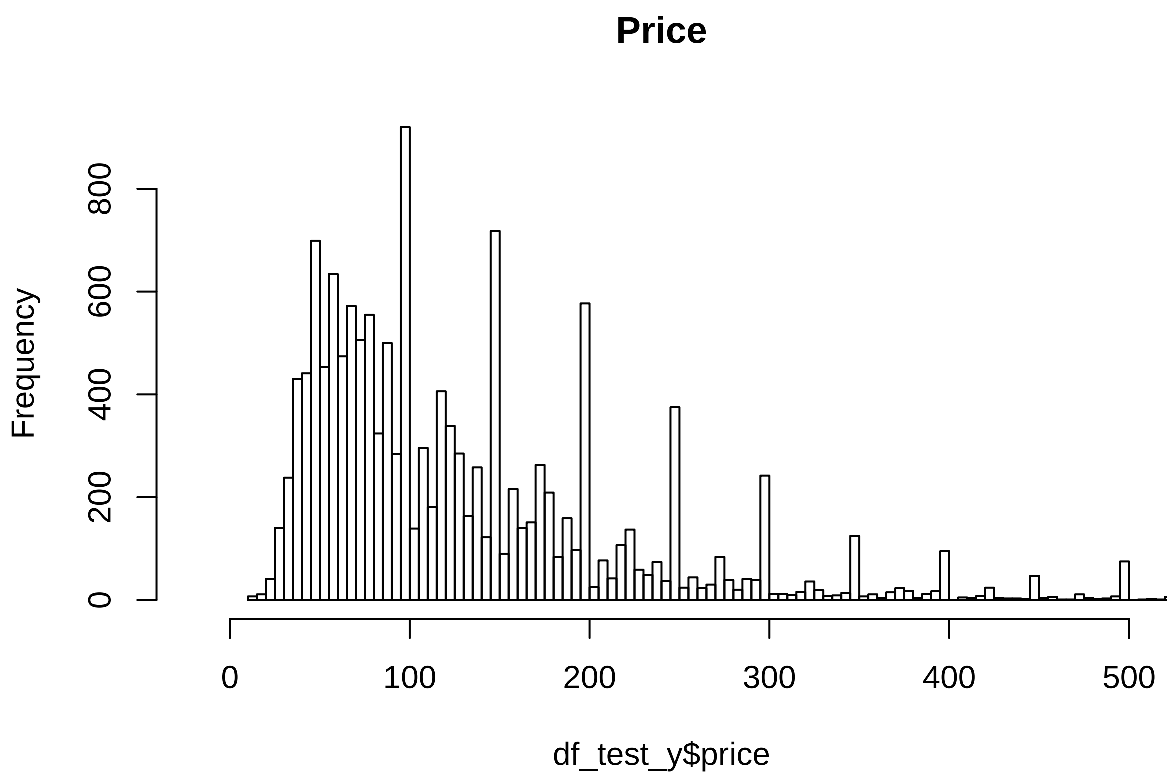
```r
# feature importance
xgb.importance(colnames(xgb.train), model = xgb.fit)
```

```
##                             Feature         Gain        Cover    Frequency
##  1:      room_type.Entire home/apt 4.258356e-01 2.000011e-01 0.0337875887
##  2:                      longitude 1.895041e-01 2.502868e-01 0.2046401622
##  3:                       latitude 9.812910e-02 1.190718e-01 0.2021624057
##  4:               availability_365 8.982579e-02 1.731157e-01 0.1669106881
##  5:                 minimum_nights 7.156341e-02 1.091317e-01 0.1576754139
##  6: calculated_host_listings_count 5.877600e-02 2.013640e-02 0.0685888050
##  7:              number_of_reviews 4.146954e-02 8.326819e-02 0.0925779930
##  8:              reviews_per_month 1.231771e-02 1.040967e-02 0.0255659421
##  9:  neighbourhood_group.Manhattan 7.075697e-03 2.442729e-02 0.0130645343
## 10:     neighbourhood_group.Queens 4.096104e-03 2.864378e-05 0.0180200473
## 11:         room_type.Private room 8.382327e-04 1.010487e-02 0.0126140331
## 12:      neighbourhood_group.Bronx 5.639152e-04 1.873163e-06 0.0042797612
## 13:   neighbourhood_group.Brooklyn 4.828745e-06 1.603896e-05 0.0001126253
```

```r
# pred price distrubution comparison
xgb.pred <- predict(xgb.fit,xgb.test,reshape=T)
hist(xgb.pred, breaks = 100,main = "XGB Pred Price",xlim = c(-20 , +500))
```




```r
hist(df_test_y$price, breaks = 1000,main = "Price",xlim = c(-20 , +500))
```

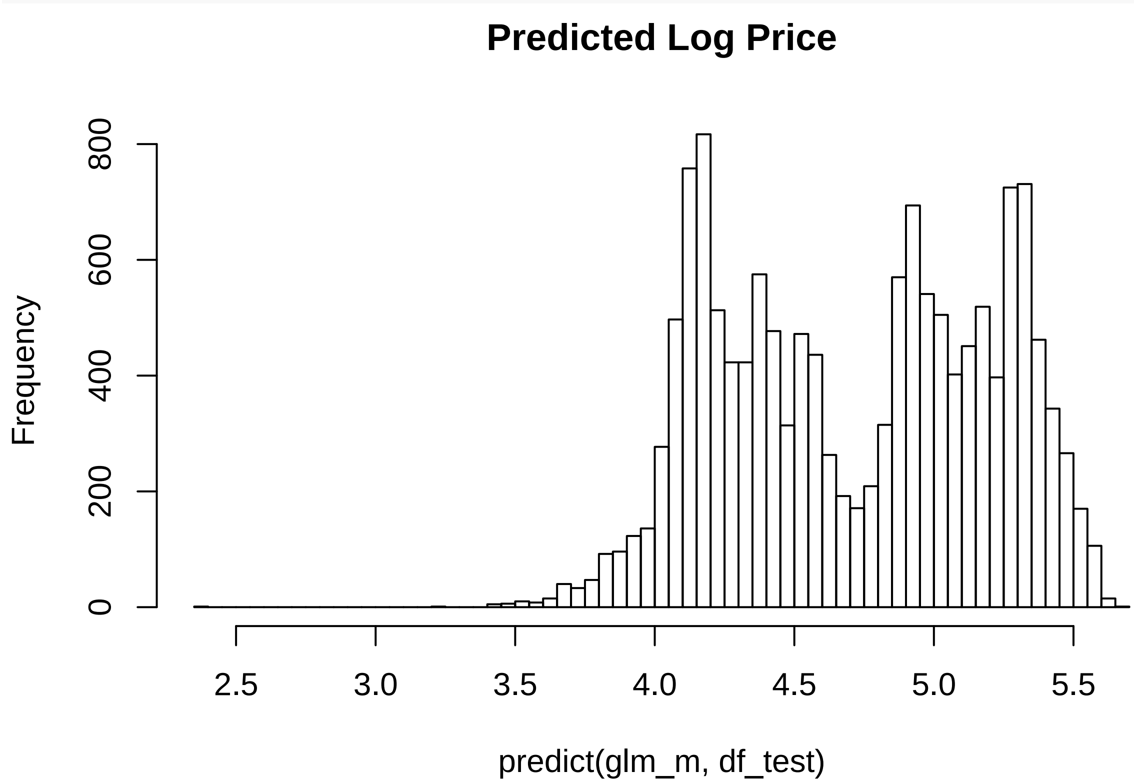



```r
# generalized linear model
df_train$logprice <- log(df_train$price)
df_test$logprice <- log(df_test$price)

glm_m <- glm(logprice ~ ., data = subset(df_train, select = -c(price)))
summary(glm_m)
```

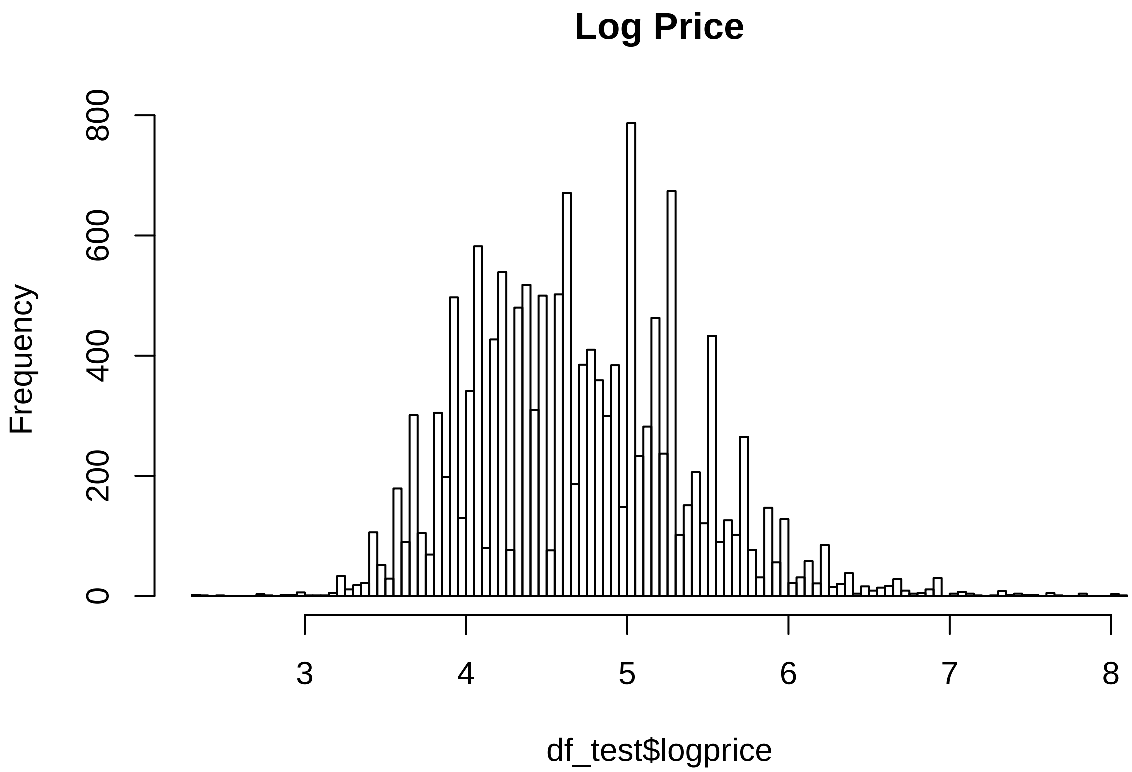
```
##
## Call:
## glm(formula = logprice ~ ., data = subset(df_train, select = -c(price)))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0250  -0.3096  -0.0483   0.2364   4.2084
##
## Coefficients:
##                                    Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                      -2.021e+02  8.168e+00  -24.746  < 2e-16
## neighbourhood_groupBrooklyn      -3.144e-02  2.210e-02   -1.422    0.155
## neighbourhood_groupManhattan      2.647e-01  2.001e-02   13.228  < 2e-16
## neighbourhood_groupQueens         8.832e-02  2.113e-02    4.179 2.93e-05
## neighbourhood_groupStaten Island -8.078e-01  4.239e-02  -19.055  < 2e-16
## latitude                         -5.595e-01  7.944e-02   -7.044 1.91e-12
## longitude                        -3.108e+00  9.188e-02  -33.824  < 2e-16
## room_typePrivate room            -7.469e-01  5.492e-03 -135.996  < 2e-16
## room_typeShared room             -1.153e+00  1.732e-02  -66.538  < 2e-16
## minimum_nights                   -2.059e-03  1.355e-04  -15.190  < 2e-16
## calculated_host_listings_count   -1.273e-04  8.017e-05   -1.587    0.112
## availability_365                  6.915e-04  2.152e-05   32.124  < 2e-16
## number_of_reviews                -8.263e-04  7.135e-05  -11.581  < 2e-16
```

```
## reviews_per_month                  1.178e-02  2.184e-03    5.393 6.98e-08
##
## (Intercept)                       ***
## neighbourhood_groupBrooklyn
## neighbourhood_groupManhattan      ***
## neighbourhood_groupQueens         ***
## neighbourhood_groupStaten Island ***
## latitude                          ***
## longitude                         ***
## room_typePrivate room             ***
## room_typeShared room              ***
## minimum_nights                    ***
## calculated_host_listings_count
## availability_365                  ***
## number_of_reviews                 ***
## reviews_per_month                 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2348707)
##
##     Null deviance: 16043.0  on 34166  degrees of freedom
## Residual deviance:  8021.5  on 34153  degrees of freedom
## AIC: 47479
##
## Number of Fisher Scoring iterations: 2
```

```
hist(predict(glm_m,df_test), breaks = 100,main = "Predicted Log Price")
```



```r
hist(df_test$logprice, breaks = 100, main = "Log Price")
```




```r
# both models are pretty good starts
# xgboost seems better based on the distribution of the residuals
# similar feature importance
```