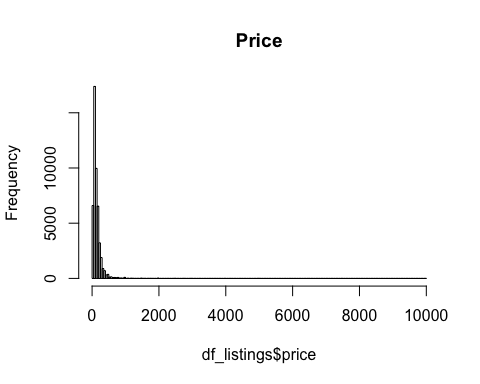
Questions

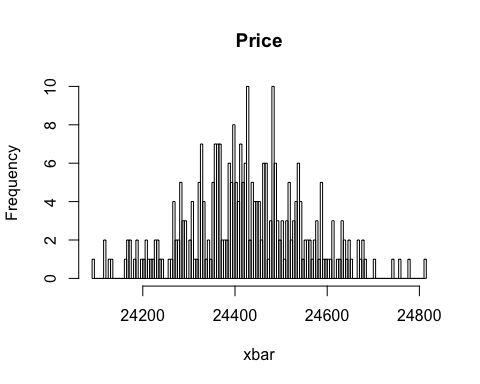
1. Please describe the central limit theorem and provide an example.

The central limit theorem tells us that when we sample from a population multiple times, despite the sample distributions, the means of these samples will be normally distributed with a mean that approaches the population mean the more times you sample. From this you can assume that the average of the sample means is the population mean which is essential for the theory behind statistical modeling. Here is a mock up example from the data provided. The price column seems to have a right skewed distribution. However, when I take an average of the means of 300 random samples of 10,000 observations, they are clearly normally distributed.

**library**(readr)  
df\_listings <- **read\_csv**("/Users/cohean/Desktop/DataSciChallenge/listings.csv",   
                       col\_types = **cols**(host\_id = **col\_character**(),   
                                        id = **col\_character**()))  
  
**hist**(df\_listings**$**price, breaks = 250, main = "Price")



n <- 300  
samplesize <- 10000  
xbar <- **rep**(NA,n)  
**for**(i **in** 1**:**n){  
 dfsamp <- **sample**(1**:nrow**(df\_listings),size=samplesize)  
 xbar[i] <- **mean**(dfsamp)   
}  
**hist**(xbar, breaks = 250, main = "Price")



1. Describe a classification algorithm that you have previously put into production and why it was chosen.

I recently worked on a project with other data scientists modeling twitter data to predict product safety issues. In order to train the model, we constantly labeled tweets, classifying them as a safety issue vs other issues. In the end, we build a pipeline that constantly queries tweets, predicts their relevancy, and report the results with a dashboard on a web server. Now we have warnings for things like car recalls and exploding phone batteries to inform the public about and potentially avoid accidents and injuries.

1. Describe the difference between bagging and boosting methods, and when to use one or the other.

Many modeling techniques use a combination of multiple weaker models, fitted using random samples from the data, that are combined to produce a stronger predictive model. Bagging and Boosting are two different methods of sampling for the multiple models that make up the final ensemble. Although both methods sample with replacement, for Bagging, each random sample of data is truly random while for Boosting, the iterative samples are weighted based on the models before them. Basically, the algorithm chooses the sample based on learning requirements at each iteration.

1. Describe 2 regularization techniques for a random forest model

Tuning parameters such as the number of trees, and the number of allowable features for each tree are both very useful ways of avoiding overfitting.

Data Challenge

Attached you will find the file listings.csv which provides data on around 50,000 AirBnB listings in New York City.  Given only this data, you want to create a model to predict how much you can charge for new listings while keeping vacancy down.

Please put together a brief analysis of the dataset and show how you would go about creating a model to predict a listing price, while taking market demand into account.

After some initial data exploration, I fitted a couple of regression models that predict the price based on the variables provided. I fit an XGBoost model and a generalized linear model which could use some tuning and feature engineering. After that I would think about launching an interactive web application in order to create a tool for users to calculate the market value of their rental. This could help keep prices strategic and ultimately decrease vacancy. Here is an R Markdown output of the analysis.

Predicting Air BnB Prices

Andrew Cohen

1/10/2020

## R Markdown

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

library(xgboost)

##   
## Attaching package: 'xgboost'

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

library(dummies)

## dummies-1.5.6 provided by Decision Patterns

library(rsample)

## Loading required package: tidyr

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)  
  
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

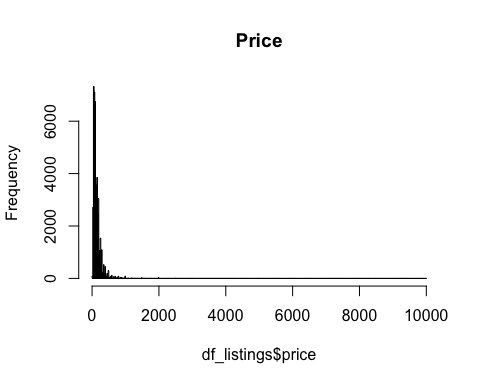
# remove nuisance columns  
# additional analysis thoughts, potentially can include the "name" column using nlp techniques  
df\_listings <- subset(df\_listings, select = -c(id, name, host\_id, host\_name, neighbourhood))  
  
# check categorical for errors vars before encoding   
df\_listings %>% count(neighbourhood\_group, 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

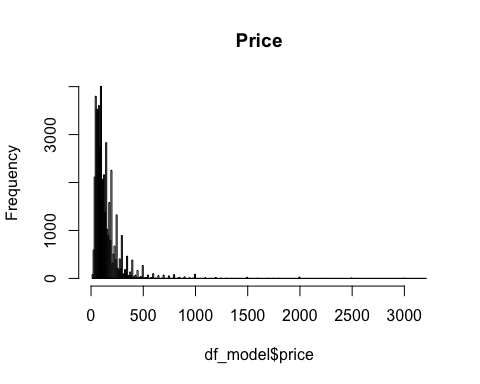
df\_listings %>% count(room\_type, sort = TRUE)

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

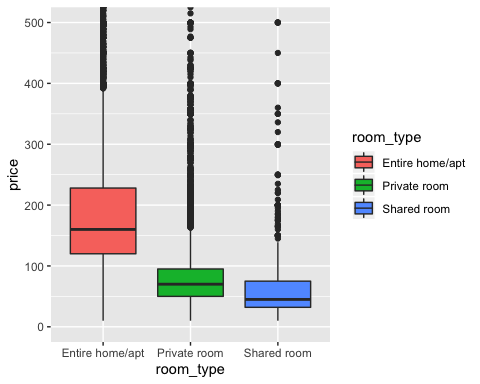
#df\_listings %>% count(neighbourhood, sort = TRUE)  
  
  
# lapply(df\_listings,class)  
hist(df\_listings$price, breaks = 500, main = "Price")



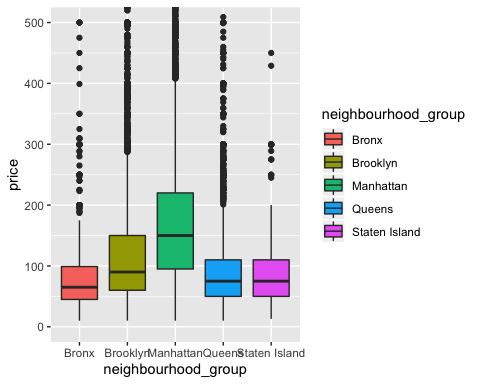
# boxplot(df\_listings$price)  
  
#remove outliers  
df\_model <- df\_listings[df\_listings$price<3500,]  
# remove zeros  
df\_model <- df\_model[df\_model$price != 0,]  
hist(df\_model$price, breaks = 250, main = "Price")



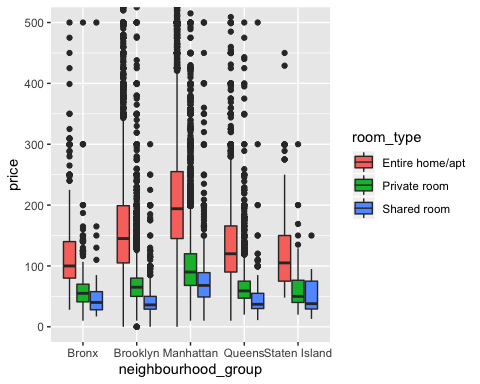
# boxplot room type  
ggplot(df\_model, aes(x=room\_type, y=price, fill=room\_type)) +   
 geom\_boxplot() +   
 coord\_cartesian(ylim = c(0, 500)) #zoom into center of data



# 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

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

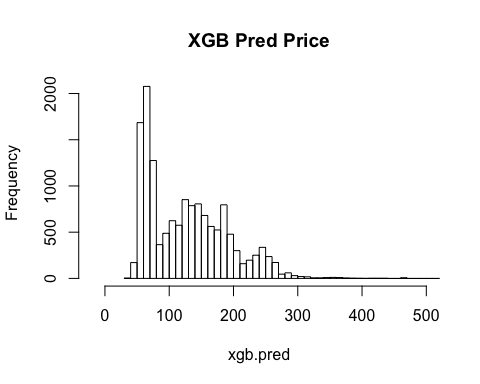
# seperate X and Y matrices  
df\_train\_x <- subset(df\_dummy\_train, select = -c(price))  
df\_train\_y <- subset(df\_dummy\_train, select = c(price))  
df\_test\_x <- subset(df\_dummy\_test, select = -c(price))  
df\_test\_y <- subset(df\_dummy\_test, select = c(price))  
  
# using xgboost is one of the best places to start for a predictive model because it usually fits very well without much tuning.  
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: 684 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

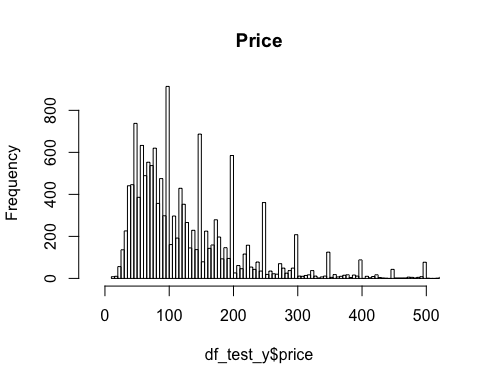
# feature importance  
xgb.importance(colnames(xgb.train), model = xgb.fit)

## Feature Gain Cover Frequency  
## 1: room\_type.Entire home/apt 0.4218285084 1.997174e-01 0.03362475  
## 2: longitude 0.1815729138 2.330068e-01 0.18527236  
## 3: latitude 0.1119538120 1.217534e-01 0.20836135  
## 4: availability\_365 0.0920004455 1.673531e-01 0.16072629  
## 5: minimum\_nights 0.0746621677 1.106368e-01 0.16565792  
## 6: number\_of\_reviews 0.0541606471 9.390369e-02 0.09683927  
## 7: calculated\_host\_listings\_count 0.0404913949 2.195000e-02 0.06624075  
## 8: reviews\_per\_month 0.0096605850 1.004297e-02 0.02824479  
## 9: neighbourhood\_group.Manhattan 0.0050707230 2.606892e-02 0.01087200  
## 10: neighbourhood\_group.Queens 0.0047920200 1.488781e-05 0.01882986  
## 11: neighbourhood\_group.Brooklyn 0.0014684083 6.585368e-05 0.00493163  
## 12: room\_type.Private room 0.0014411331 1.548383e-02 0.01457072  
## 13: neighbourhood\_group.Bronx 0.0008972411 2.360976e-06 0.00582829

# 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))



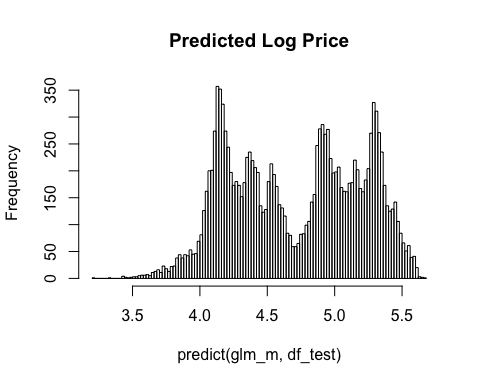
hist(df\_test\_y$price, breaks = 1000,main = "Price",xlim = c(-20 , +500))



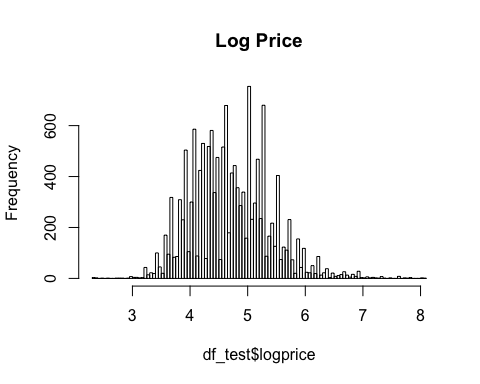
# generalized linear model  
df\_train$logprice <- log(df\_train$price)  
df\_test$logprice <- log(df\_test$price)  
  
glm\_m <- glm(logprice ~ ., data = subset(df\_train, select = -c(price)))  
summary(glm\_m)

##   
## Call:  
## glm(formula = logprice ~ ., data = subset(df\_train, select = -c(price)))  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.0303 -0.3067 -0.0499 0.2387 4.1952   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -1.959e+02 8.183e+00 -23.946 < 2e-16  
## neighbourhood\_groupBrooklyn -7.376e-02 2.234e-02 -3.302 0.000961  
## neighbourhood\_groupManhattan 2.246e-01 2.024e-02 11.098 < 2e-16  
## neighbourhood\_groupQueens 4.181e-02 2.146e-02 1.949 0.051354  
## neighbourhood\_groupStaten Island -8.739e-01 4.218e-02 -20.722 < 2e-16  
## latitude -6.398e-01 7.971e-02 -8.027 1.03e-15  
## longitude -3.069e+00 9.186e-02 -33.413 < 2e-16  
## room\_typePrivate room -7.489e-01 5.506e-03 -136.027 < 2e-16  
## room\_typeShared room -1.156e+00 1.743e-02 -66.288 < 2e-16  
## minimum\_nights -1.998e-03 1.299e-04 -15.378 < 2e-16  
## calculated\_host\_listings\_count -1.030e-04 7.862e-05 -1.310 0.190162  
## availability\_365 7.253e-04 2.143e-05 33.844 < 2e-16  
## number\_of\_reviews -8.678e-04 6.969e-05 -12.452 < 2e-16  
## reviews\_per\_month 1.086e-02 2.069e-03 5.250 1.53e-07  
##   
## (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 0.2351061)  
##   
## Null deviance: 16117.4 on 34166 degrees of freedom  
## Residual deviance: 8029.6 on 34153 degrees of freedom  
## AIC: 47514  
##   
## Number of Fisher Scoring iterations: 2

hist(predict(glm\_m,df\_test), breaks = 100,main = "Predicted Log Price")



hist(df\_test$logprice, breaks = 100, main = "Log Price")



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