Similary and Emsemble Part 1

2023-03-24

#Load vcd and ISLR

```
library(ISLR)
```

#load dataset

```
df <- read.csv("train.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"

df <- df[,c(2,6,7,24,25,28,29)]
names(df)

## [1] "log_price"            "accommodates"         "bathrooms"
## [4] "number_of_reviews"    "review_scores_rating" "bedrooms"
## [7] "beds"

df <- na.omit(df)
dim(df)

## [1] 57129     7

summary(df)

##    log_price      accommodates      bathrooms     number_of_reviews
##  Min.   :0.000   Min.   : 1.000   Min.   :0.000   Min.   :  1.00
##  1st Qu.:4.304   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.:  3.00
##  Median :4.700   Median : 2.000   Median :1.000   Median : 11.00
##  Mean   :4.750   Mean   : 3.223   Mean   :1.227   Mean   : 26.93
##  3rd Qu.:5.165   3rd Qu.: 4.000   3rd Qu.:1.000   3rd Qu.: 33.00
```

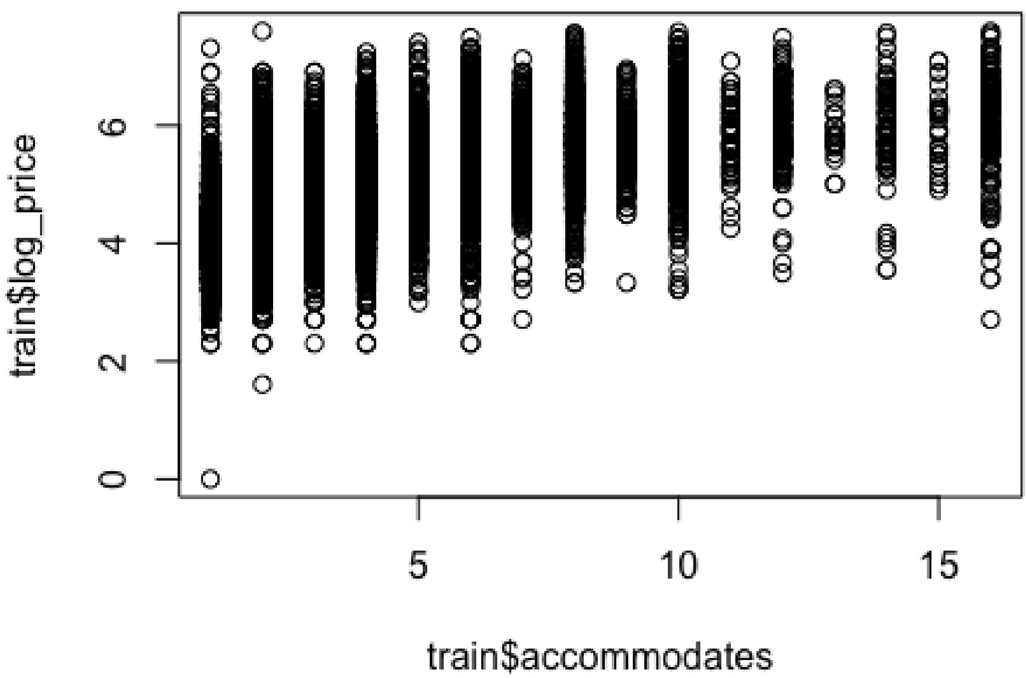
```
##  Max.   :7.600   Max.   :16.000   Max.   :8.000   Max.   :605.00
##  review_scores_rating    bedrooms            beds
##  Min.   : 20.00       Min.   : 0.000   Min.   : 0.00
##  1st Qu.: 92.00       1st Qu.: 1.000   1st Qu.: 1.00
##  Median : 96.00       Median : 1.000   Median : 1.00
##  Mean   : 94.08       Mean   : 1.262   Mean   : 1.74
##  3rd Qu.:100.00       3rd Qu.: 1.000   3rd Qu.: 2.00
##  Max.   :100.00       Max.   :10.000   Max.   :18.00
```

```
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.8, replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Plot the relationship between the price and every predictor individually to find out which predictors are unnecesary

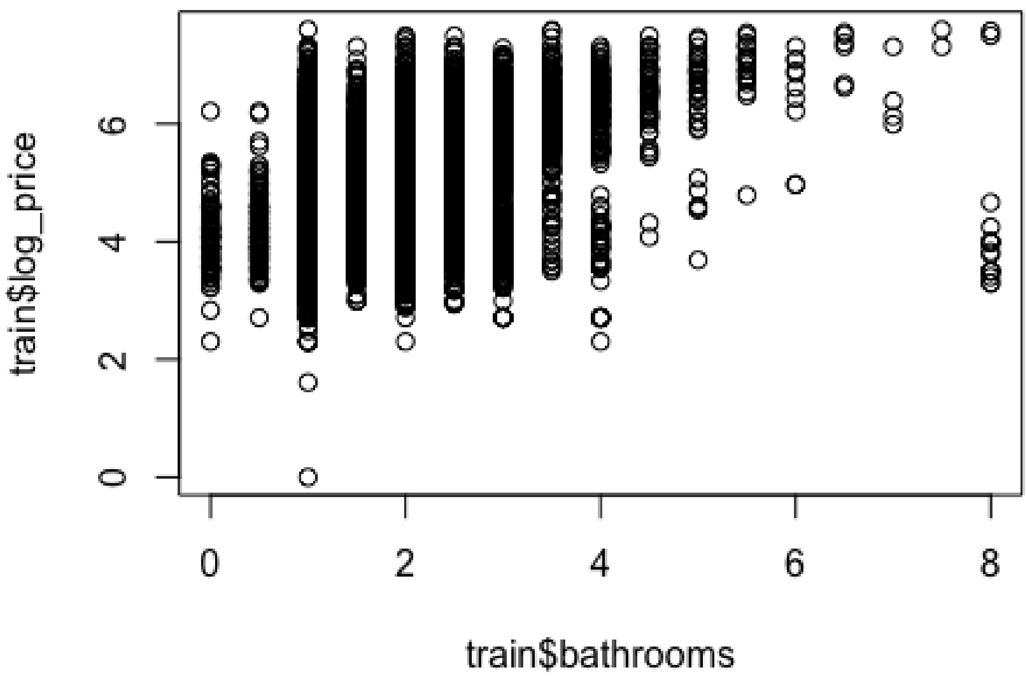
accomodates has a bit of a correlation

```
plot(train$log_price~train$accommodates)
```



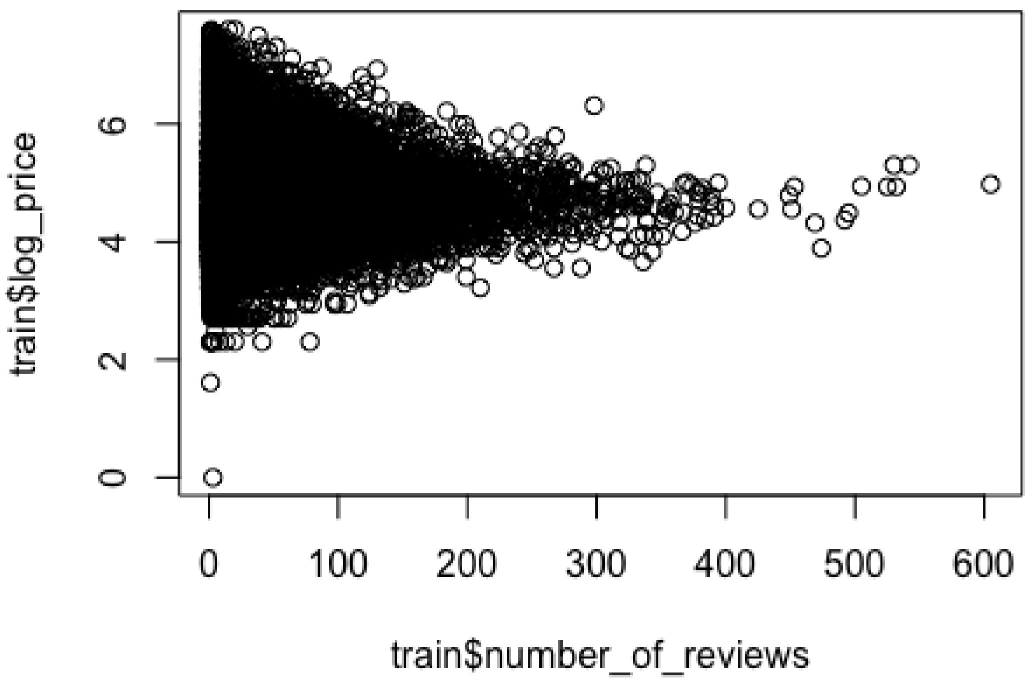
bathroom has a correlation

```
plot(train$log_price~train$bathrooms)
```



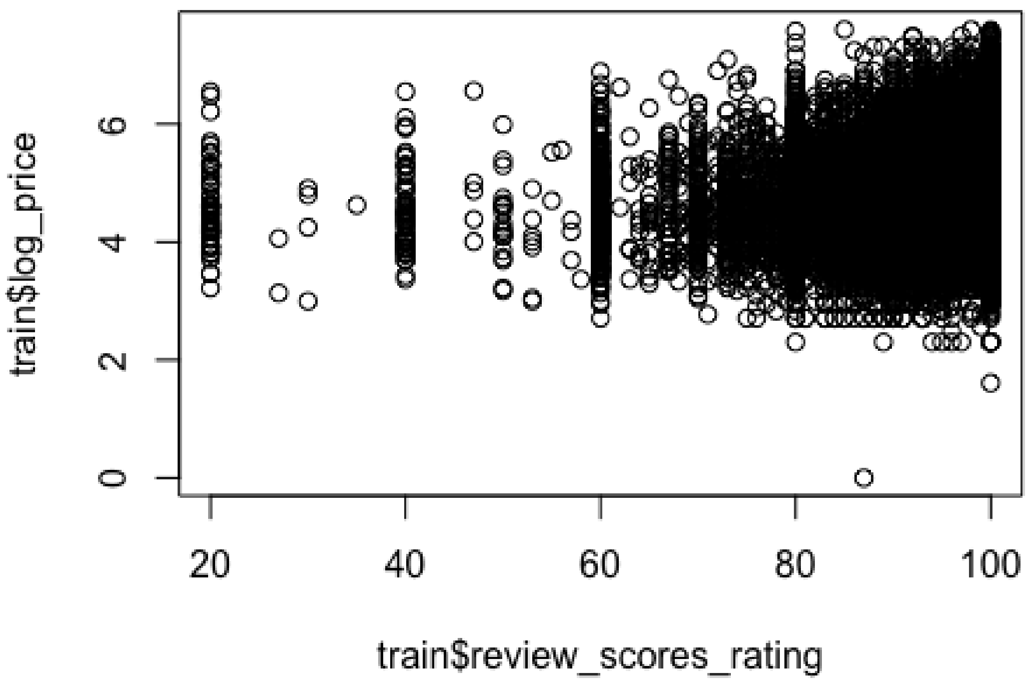
The number of reviews has no correlation for some reason

```
plot(train$log_price~train$number_of_reviews)
```



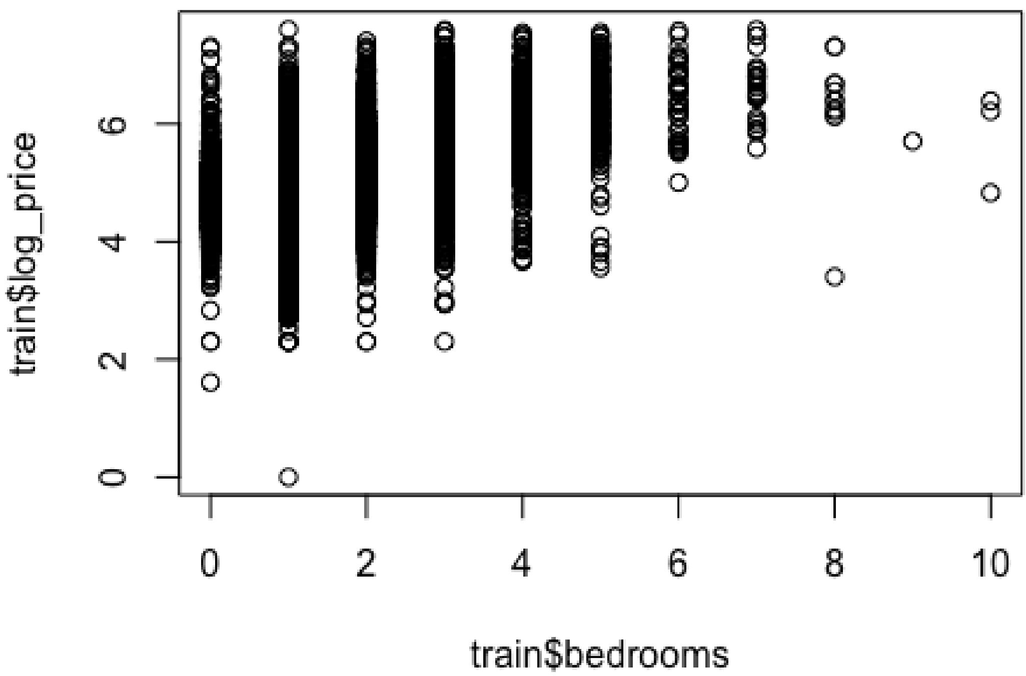
Review score rating shockingly has no correlation either

```
plot(train$log_price~train$review_scores_rating)
```



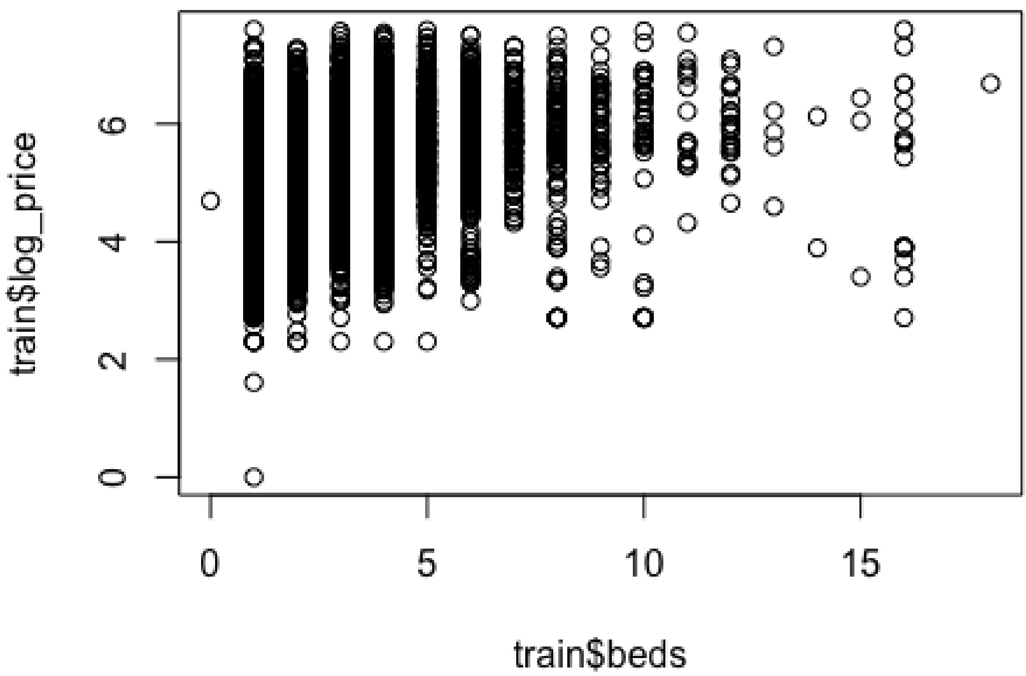
Bedrooms have a correlation

```
plot(train$log_price~train$bedrooms)
```



Number of beds also have a correlation

```
plot(train$log_price~train$beds)
```



We will use acccomodates, bathrooms, bedrooms, and be for the linear regression model

```
lm1 <-
lm(train$log_price~train$accommodates+train$bathrooms+train$bedrooms+train$be
ds)
summary(lm1)
```

```
##
## Call:
## lm(formula = train$log_price ~ train$accommodates + train$bathrooms +
##     train$bedrooms + train$beds)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3650 -0.3576  0.0045  0.3575  3.0595
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         4.079579   0.006156  662.70   <2e-16 ***
## train$accommodates  0.175920   0.002201   79.93   <2e-16 ***
## train$bathrooms     0.061804   0.005601   11.04   <2e-16 ***
## train$bedrooms      0.116617   0.004727   24.67   <2e-16 ***
## train$beds         -0.068948   0.003669  -18.79   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5371 on 45698 degrees of freedom
## Multiple R-squared:  0.3612, Adjusted R-squared:  0.3612
## F-statistic:  6461 on 4 and 45698 DF,  p-value: < 2.2e-16
```

We predict the correlation and mse of the linear regression model. They are not too great

pred1 <- predict(lm1, newdata=test)
cor_lm1 <- cor(pred1, test$log_price) ## mse_lm1 <
mean((pred1~test$logprice)^2)
print(paste("cor=", cor_lm1))
print(paste("mse=", mse_lm1))

See if we need to convert anything for the knn regression

```
str(test)

## 'data.frame':    11426 obs. of  7 variables:
##  $ log_price           : num  4.98 4.74 4.6 4.01 5 ...
##  $ accommodates        : int  5 2 2 2 6 2 2 2 6 2 ...
##  $ bathrooms           : num  1 1 2 1 1 1 1 1 2 2 ...
##  $ number_of_reviews   : int  10 4 12 2 14 34 85 3 30 25 ...
##  $ review_scores_rating: num  92 40 88 100 100 88 96 100 89 100 ...
##  $ bedrooms            : num  1 0 1 1 3 1 1 1 2 1 ...
##  $ beds                : num  3 1 1 1 3 1 1 1 2 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:16982] 4 13 16 25 32 34 35 39
41 42 ...
##   ..- attr(*, "names")= chr [1:16982] "4" "13" "16" "25" ...
```

Import the libraries needed for knn regression

```
library(ggplot2)
library(caret)

## Loading required package: lattice
```

Use knn regression to predict the values in the test data

fit <- knnreg(train[,2:7],train[,1],k=3)
pred2 <- predict(fit, test[,2:7])
cor_knn1 <- cor(pred2, test$log_price) ## mse_knn1 <- mean((pred2-test$log_price)^2)
print(paste("cor=", cor_knn1))
print(paste("mse=", mse_knn1))

knn does not know how to handle ties in distance unfortunately

Scaling should lead to more accurate results

```
train_scaled <- train[,2:7]
means <- sapply(train_scaled, mean)
```

```
stdvs <- sapply(train_scaled, sd)
train_scaled <- scale(train_scaled, center=means, scale=stdvs)
test_scaled <- scale(test[,2:7], center=means, scale=stdvs)
```

this is used to find the best k value to use for the knn regression

cor_k <- rep(0,20)
mse_k <- rep(0,20)
i <- 1
for (k in seq(1,39,2)){
fit_k <- knnreg(train_scaled, train$log_price, k=k) ## pred_k <- predict(fit_k, test_scaled) ## cor_k[i] ## <- cor(pred_k, test$log_price)
mse_k[i] <- mean((pred_k - test$log_price)^2)
print(paste("k=", k, cor_k[i], mse_k[i]))
i <- i+1
}

plot(1:20, cor_k, lwd=2, col='red', ylab='', yaxt='n') par(new=TRUE) plot(1:20, mse_k, lwd=2, col='blue', labels=FALSE< ylab='', yaxt='n')

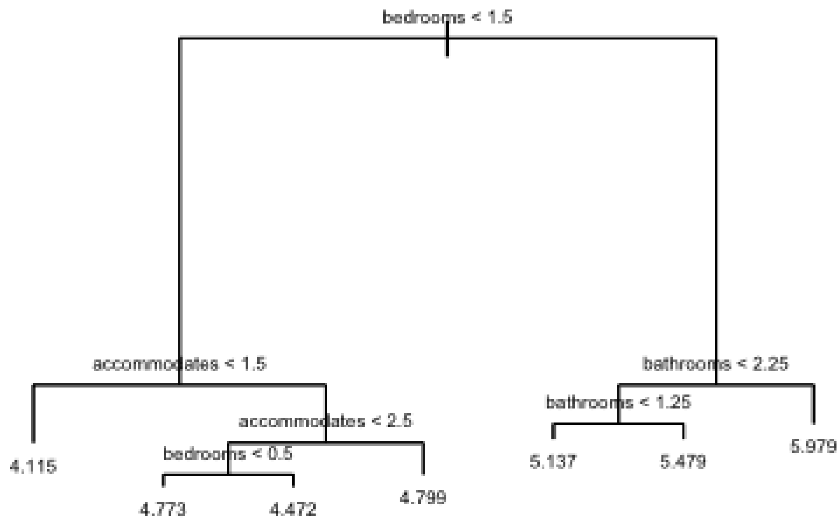
import the necessary libraries for decision trees

```
library(tree)
library(MASS)
```

Predict the correlation and rmse of the test data using decision trees

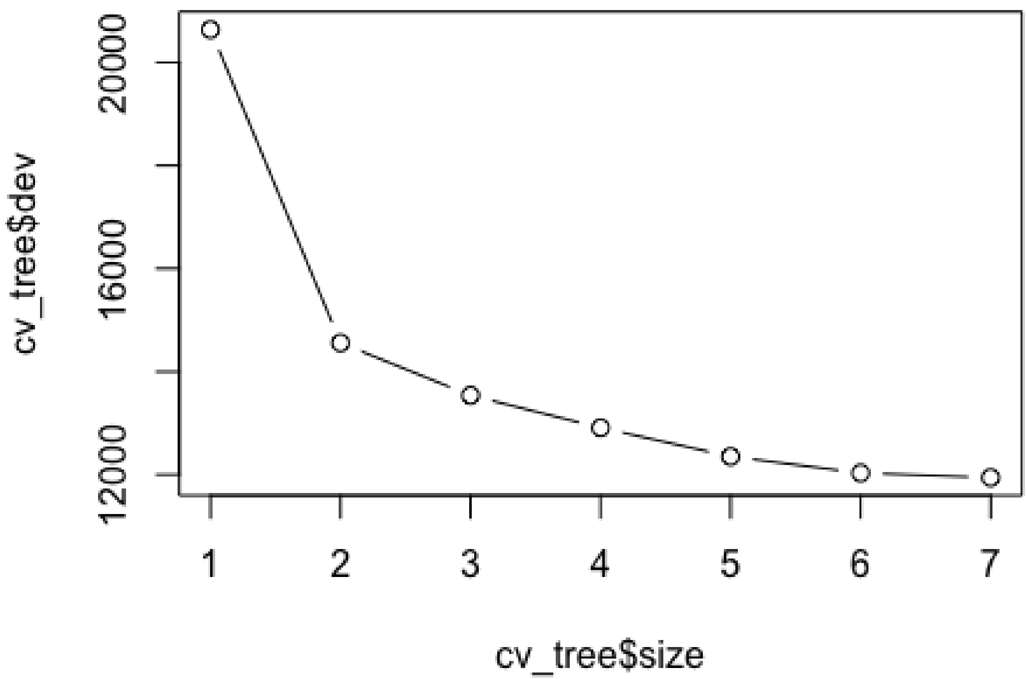
```
tree1 <- tree(train$log_price~., data=train)
summary(tree1)

##
## Regression tree:
## tree(formula = train$log_price ~ ., data = train)
## Variables actually used in tree construction:
## [1] "bedrooms"     "accommodates" "bathrooms"
## Number of terminal nodes:  7
## Residual mean deviance:  0.2588 = 11820 / 45700
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -4.115000 -0.310000 -0.006228  0.000000  0.323700  3.198000

pred <- predict(tree1, newdata=test)
print(paste('correlation:', cor(pred,test$log_price)))

## [1] "correlation: 0.649090838438933"

rmse_tree <- sqrt(mean((pred-test$log_price)^2))
print(paste('rmse:', rmse_tree))

## [1] "rmse: 0.499506789880523"
```

```
plot(tree1)
text(tree1,cex=0.5, pretty=0)
```



Use cross validation to find the best point to prune the tree

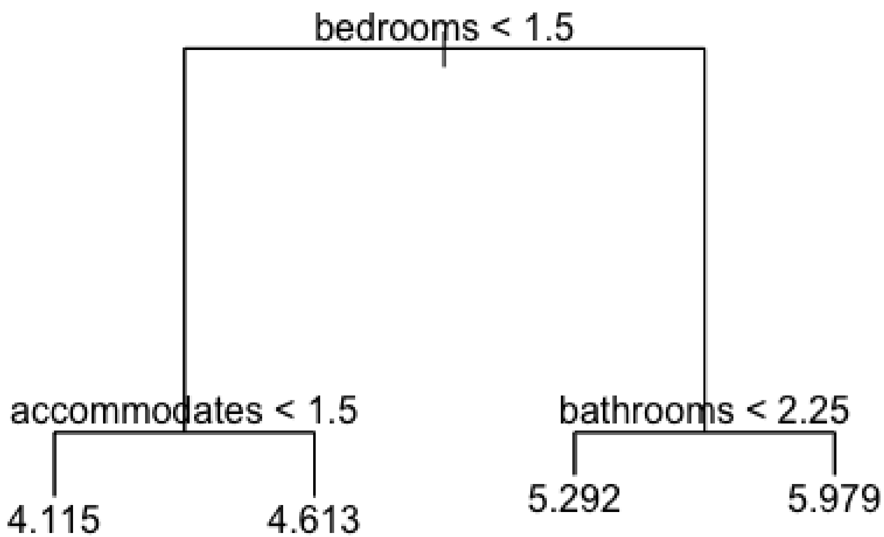
```
cv_tree <- cv.tree(tree1)
plot(cv_tree$size, cv_tree$dev, type='b')
```



The best value seems to be 4

Prune the tree

```
tree_pruned <- prune.tree(tree1, best=4)
plot(tree_pruned)
text(tree_pruned,pretty=0)
```



test for the correlation and rmse on the pruned tree. The results seemed to have slightly improved

```
pred_pruned <- predict(tree_pruned, newdata=test)
print(paste('correlation:', cor(pred_pruned,test$log_price)))

## [1] "correlation: 0.610891094691187"

rmse_pruned <- sqrt(mean((pred_pruned-test$log_price)^2))
print(paste('rmse:', rmse_pruned))

## [1] "rmse: 0.519814786954771"
```

Use a Random Forest to counter high variance

```
library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1234)
rf <- randomForest(train$log_price~., data=train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = train$log_price ~ ., data = train, importance =
TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.2333011
##                     % Var explained: 48.33
```

test for correlation and rmse on random forest

```
pred_rf <- predict(tree_pruned, newdata=test)
print(paste('correlation:', cor(pred_rf,test$log_price)))
```

```
## [1] "correlation: 0.610891094691187"
```

```
rmse_rf <- sqrt(mean((pred_rf-test$log_price)^2))
print(paste('rmse:', rmse_rf))
```

```
## [1] "rmse: 0.519814786954771"
```

The result are the same.

The linear regression model simply draws a line that best follows the slope of the price versus the predictors. Since the graph is not linear at all, linear regression is not the best choice for the dataset.

The knn regression model finds the k closest point within a certain distance of a point on the graph and uses those points to predict what the price would be of an Airbnb at that point. Since the points are spread out but tend to be in groups with mostly the same of one predictor, I believe knn regression would be best for this dataset.

The decision tree just draws a line to separate the points and its a greedy algorithm so it does not go back to correct its mistakes. Because of all this, the decision tree tends to not be the best choice for accuracy, but it is much easier to interpret.