AD699 B1
Yuanjie Li, Lin Li, Hang Chang, Ziling Liu, Yini Feng, Xin Li
Due Date:05/06/2019

Semester Project: AD699

Step I: Data Preparation & Exploration (15 points)

I. Missing Values

A. Does your data contain any missing values and/or blank cells? If so, what can you do about this? Show the R code that you used to handle your missing values. Write one paragraph describing what you did, and why.

Our city is San Francisco. After filtering the dataset, we can get the subset only containing the Airbnb data of San Francisco, which has 6434 rows and 29 columns. In the subdataset, there are many missing values and blank cells. In terms of missing values, the column of bathrooms has 17 NAs. The bedrooms column has 6 NAs. The beds column has 11 NAs and the review_scores_rating column has 1387 NAs. With respect to blank cells, there are 10 columns containing them. The most blank cells are in columns host_response_rate which is 1989 and first_review as well as last_review both are 1317. If I drop the rows contains NA, the dataset will decrease 21.55%(1387/6434) at least. It is not beneficial for us to conduct data mining. Due to these columns are numerical, I replace the NA with their medians. For the blank cells, most of the columns are date, factor or zip code. In our further analysis, we may use them or not. In this step, we replace them with NA. If we utilize them in the future, we will handle them.

```
library(readr)
airbnb<-read.csv("airbnbtrain.csv")
View(airbnb)
library(dplyr)
bnbSF<-filter(airbnb, city=="SF")
View(bnbSF)
bnbSF[bnbSF==""] <-NA
anyNA(bnbSF)
summary(bnbSF)
median(bnbSF$bathrooms,na.rm = TRUE)
```

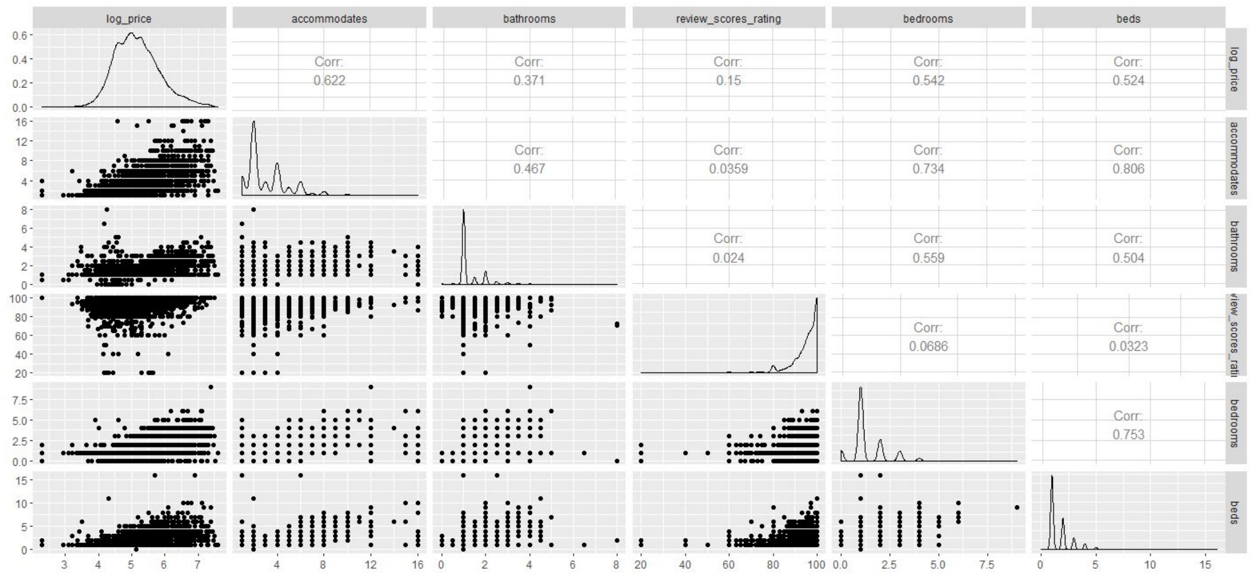
```
bnbSF$bathrooms[is.na(bnbSF$bathrooms)]<-median(bnbSF$bathrooms,na.rm =
TRUE)
anyNA(bnbSF$bathrooms)
bnbSF$bedrooms[is.na(bnbSF$bedrooms)]<-median(bnbSF$bedrooms,na.rm = TRUE)
anyNA(bnbSF$bedrooms)
bnbSF$beds[is.na(bnbSF$beds)]<-median(bnbSF$beds,na.rm = TRUE)
anyNA(bnbSF$beds)
bnbSF$review_scores_rating[is.na(bnbSF$review_scores_rating)]<-median(bnbSF$revi
ew_scores_rating,na.rm = TRUE)
```

II. Summary Statistics

A. Choose any five of the summary statistics functions shown in the textbook, class slides, or anywhere else to learn a little bit about your data set.
B. Show screenshots of the results. Describe your findings in 1-2 paragraphs.

1.

```
library(GGally)
bnbSFsub<-select(bnbSF,log_price,accommodates,bathrooms,review_scores_rating,be
drooms,beds)
ggpairs(bnbSFsub)
```



2.

```
> fivenum(bnbSF$bathrooms)
[1] 0.0 1.0 1.0 1.5 8.0
```

3.

```
range(bnbSF$bedrooms)
[1] 0 9
sd(bnbSF$bedrooms)
[1] 0.8794001
```

4.

```
library(e1071)
> skewness(bnbSF$log_price)
[1] 0.4506222
> kurtosis(bnbSF$log_price)
[1] 0.4498561
```

5.

```
table(bnbSF$room_type)
Entire home/apt    Private room     Shared room
         3818            2518             98
```

With respect to the numerical variables, we can calculate their correlations. We can find that the variable accommodates has the largest correlation 0.622 with log_price compared with other four variables. The correlation between review_scores_rating and log_price is the smallest 0.15 in the first row. From the first row, we can conclude the strength of the correlation between log_price and other variables, which lay a solid foundation for further data mining.

The fivenum function gives the values of minimum, lower-hinge, median, upper-hinge, maximum. From the fivenum result, we can infer that most of bedrooms are less than 2 and the extreme values like 8 exist.

The function of range returns a vector containing the minimum and maximum of bedrooms and sd is to calculate the standard deviation of rooms. From the results, we can find that the values of bedrooms are concentrated due to the fact that the range is large 0 to 9 and the standard deviation is relatively small 0.8794001.

Skewness is a measure of the asymmetry of the probability distribution. The skewness of log_price is 0.4506222, which is positive, and the distribution of log_price is said to be right-tailed, which means the mass of the distribution of log_price is concentrated on the left of the figure. Kurtosis is a measure of the "tailedness" of the probability distribution. The kurtosis of any univariate normal distribution is 3. It is common to compare the kurtosis of a distribution to this value. The kurtosis is 0.4498561, which is less than 3. It means that the distribution of log_price produces fewer and less extreme outliers than does the normal distribution.

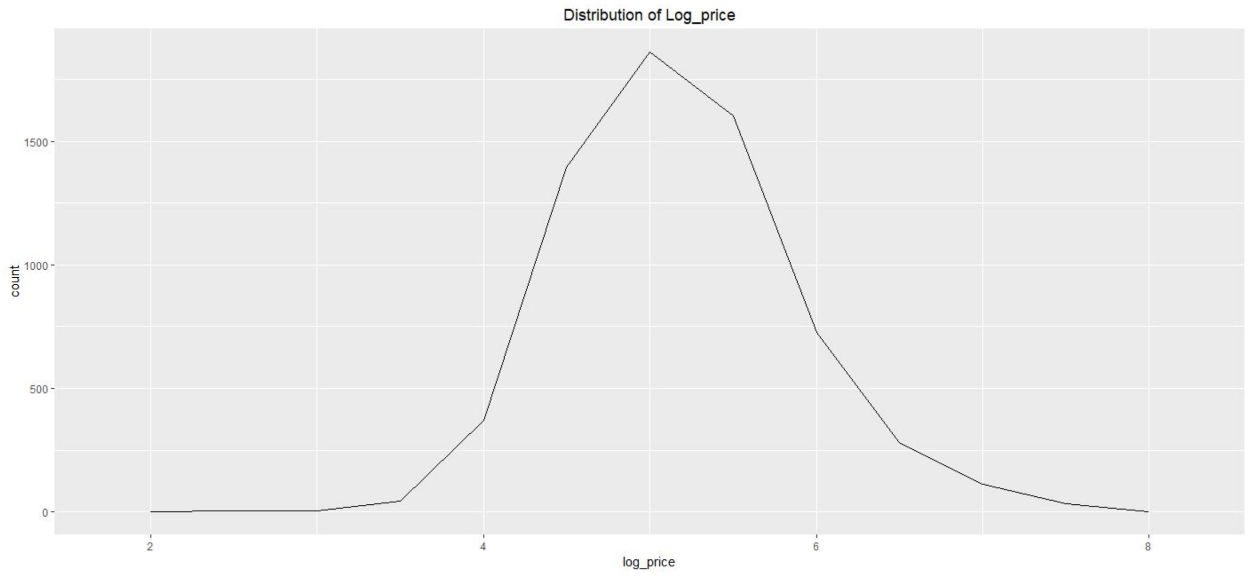
The function table uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. From the table function, we can indicate that there are 3818 entire home/apts, 2518 private rooms and 98 shared rooms.

<u>III. Visualization</u>

A. Using ggplot, create any five plots that help to describe your data.
B. Write a two-paragraph description that explains the choices that you made, and what the resulting plots show.

1.

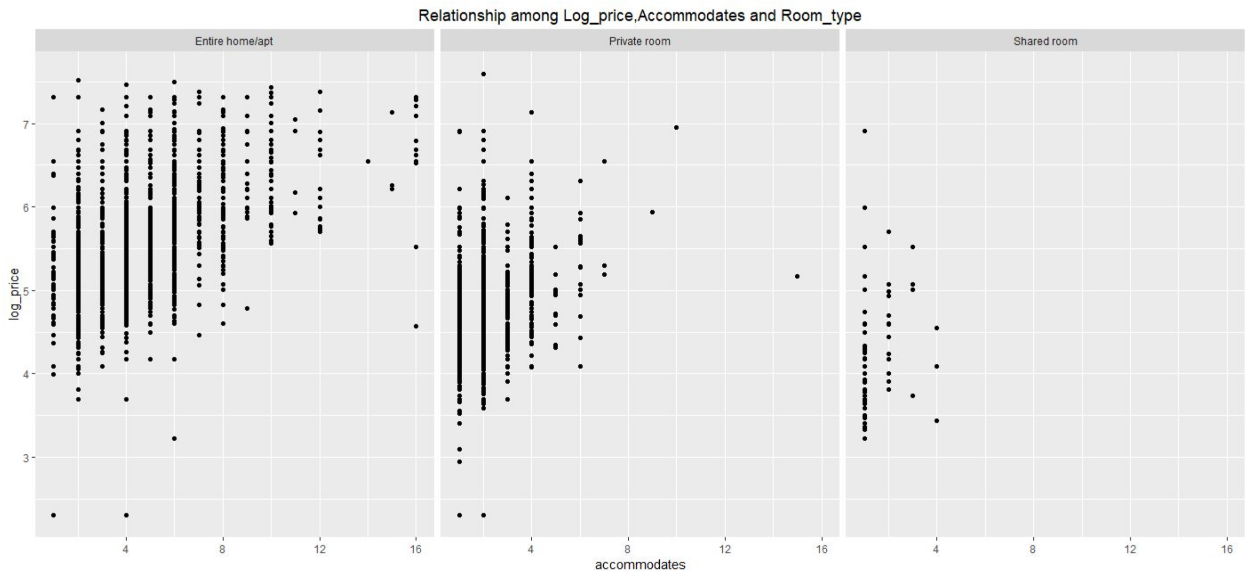
```
library(ggplot2)
p<-ggplot(data=bnbSF,aes(x=log_price)) +
geom_freqpoly(binwidth=0.5)+ggtitle("Distribution of Log_price")+theme(plot.title =
element_text(hjust = 0.5))
```



I choose log_price due to the fact that it will become the dependent variable in the following question. We should understand the attributes of it, such as distribution. In this graph, we use Frequency Polygon to show the distribution of log_price.We can find that the distribution approximates to normal distribution. The mass of the distribution of log_price is concentrated on the left of the figure. The distribution of log_price produces fewer and less extreme outliers than does the normal distribution.

2.

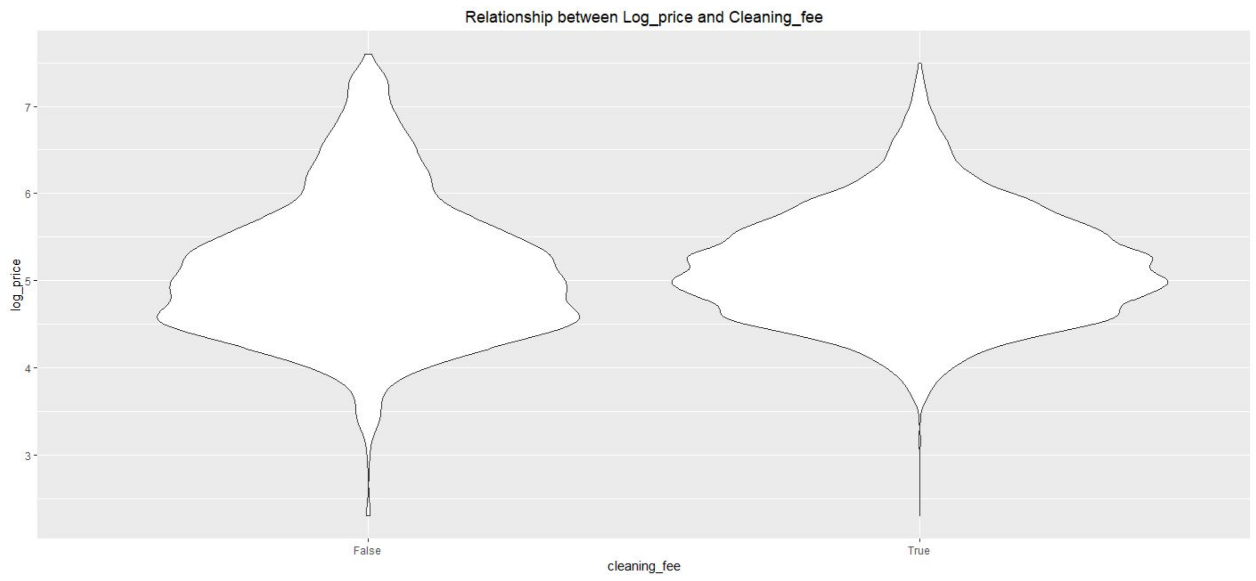
```
q <- ggplot(bnbSF, aes(x=accommodates, y=log_price)) + geom_point()
+facet_grid(bnbSF$room_type)+ggtitle("Relationship among Log_price,Accommodates
and Room_type")+theme(plot.title = element_text(hjust = 0.5))
```



I want to know the relationship between log_price and room_type and at the same I am interested in the connection between room_type and accommodates. The first relationship can give us a general picture between log_price and room_type, which is helpful for the following analysis. The graph shows that the high log_price is concentrated in Entire home/apt, which indicates room_type may be an important factor for log_price. On the other hand, Entire home/apt is more popular when the number of accommodates is large.

3.

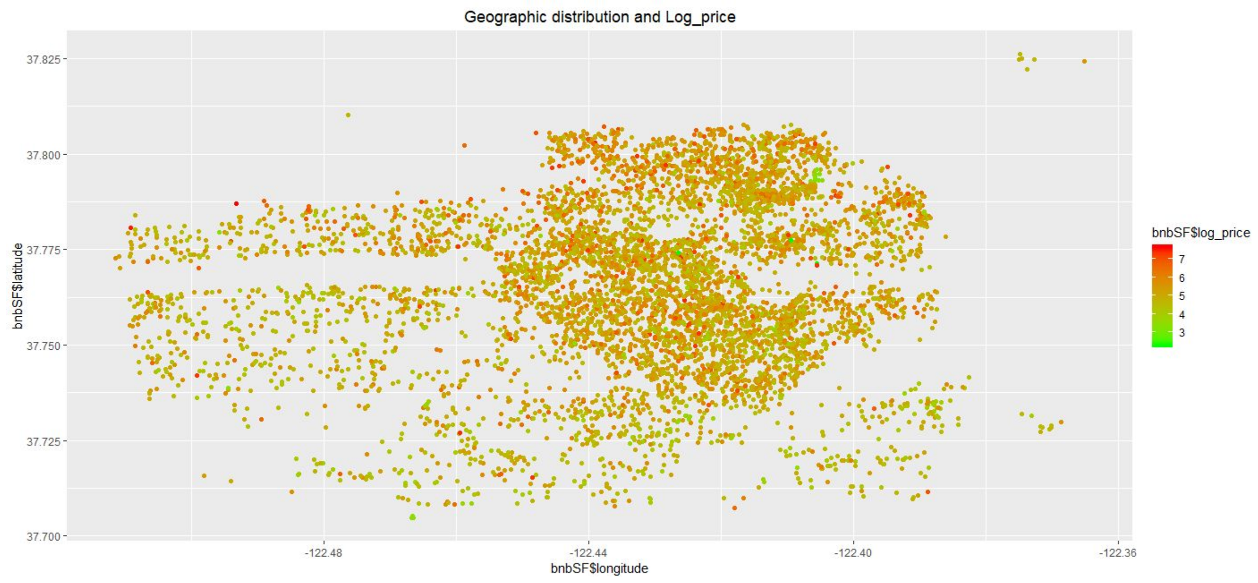
```
x <- ggplot(bnbSF, aes(x=cleaning_fee, y=log_price))
+geom_violin()+ggtitle("Relationship between Log_price and
Cleaning_fee")+theme(plot.title = element_text(hjust = 0.5))
```



I want to know whether cleaning_fee is a determinant of log_price. So I choose log_price and cleaning_fee to draw violin plots. From the graph, we can see that the distributions of log_price are similar when the cleaning_fee is true or false. This means that cleaning_fee has little influence on log_price.

4.

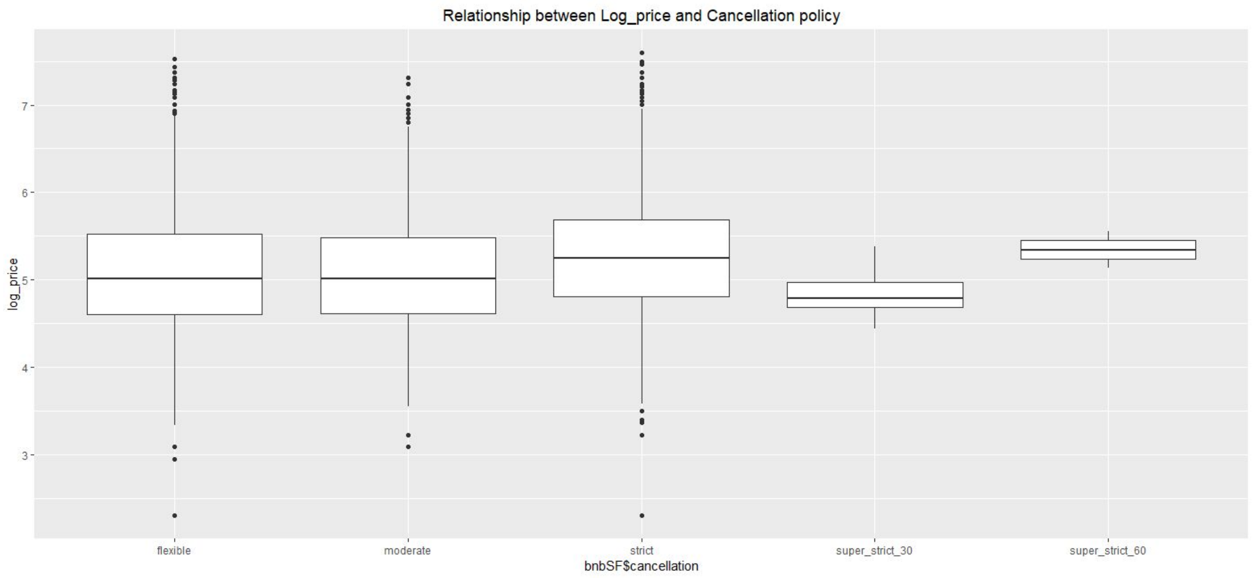
```
g<-ggplot(bnbSF,aes(x=bnbSF$longitude,y=bnbSF$latitude,color=bnbSF$log_price))+geom_point()+ggtitle("Geographic distribution and Log_price")+theme(plot.title = element_text(hjust = 0.5))
g+scale_color_gradient(low="green", high="red")
```



I want to know whether the position has an influence on log_price. From the geographic distribution above, we can conclude that the high log_price is concentrated in the middle and north. Because the red points in these areas are much more than those in other areas.

5.

```
z <- ggplot(bnbSF, aes(x=bnbSF$cancellation, y=log_price))
+geom_boxplot()+ggtitle("Relationship between Log_price and Cancellation
policy")+theme(plot.title = element_text(hjust = 0.5))
```



I am interested in the relationship between log_price and cancellation policy. So I choose them to draw a boxplots. From the graph, we can indicate that the distributions of log_price based on five different cancellation policy are very closed. In sum, cancellation policy is not a significant factor to determine log_price.

Step II: Prediction (20 points)

I. Create a multiple regression model with the outcome variable log_price .

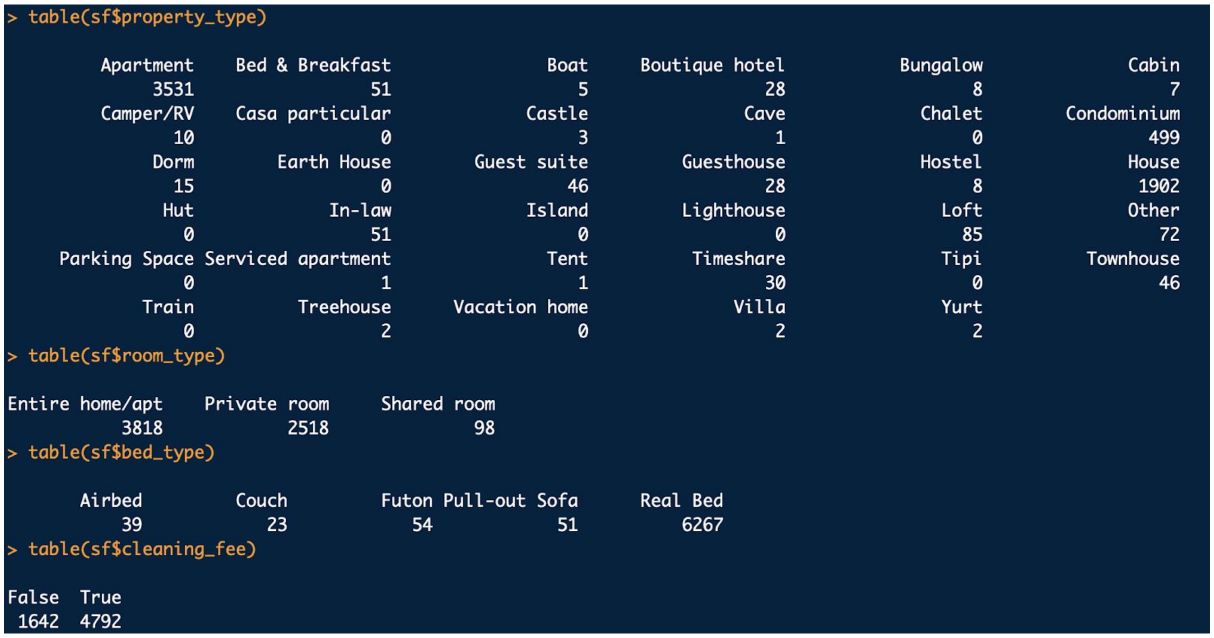
A) **Describe your process. How did you wind up including the independent variables that you kept, and discarding the ones that you didn't keep? In a narrative of at least two paragraphs, discuss your process and your reasoning. In the write-up, be sure to talk about how you evaluated the quality of your model.**

Based on the work done at the Step I, we find out that there are many predictors which are seemed to be relative to the variable log_price, such as the room_type, cleaning fee, accommodates, longitude and latitude. To some extent, those plots or mathematical statistics can reveal some relationships. For example, the more accommodates a rental can bare, the higher log_price will be. However, through the visualization, we cannot precisely measure their relationships in a quantitative way. Therefore, we want to put those variables in our multiple regression model. The first group of predictors are **room_type**, **cleaning fee**, **accommodates**, **longitude** and **latitude**. What is more, we need to find out other predictors that can also have an effect on rental price. The common knowledge instructs me that the numbers of **bedrooms**, **beds**, **bathrooms** are deterministic factors for the price, which directly determine living quality. Moreover, we think the **review_rating_score** can also reflect the quality of the rental unit. The last two predictors are **bed_type** and **property_type** because different types of bed have different price, so well as property_type. In the nutshell, we pick up these 11 independable variables.

Other variables maybe somehow important, but they are difficult to quantify, like amenities and neighbourhoods. Actually, I did think of a solution to quantify them, which is to use text mining method to identify those keywords like "Cable TV". The units with more amenities will be estimated as having a higher price. But after doing research on webs and watching some tutorial videos, it seems to be a huge project, so we give up on that. The good thing is that almost every units have similar amenities, and the estimation of neighbourhood can be replaced by longitude and latitude.

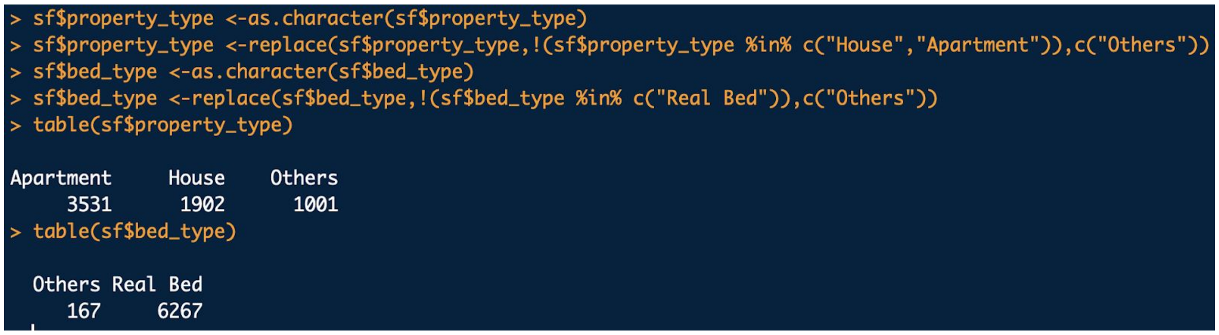
The process of building this MLR model is kind of interesting. To begin with, I have to set up a few dummy variables of which are property_type, room_type, bed_type and cleaning fee. But here is a problem. The levels of property_type and bed_type are too much, and some levels only have little amount. For example, there are 35 types of property, the Apartment and House takes most of them. Therefore, we put those types which only takes little place into one type "Others". As a result, there are only three types in property_type and two in bed_types. After setting up dummy variables, I bind all variables into a new dataframe “regdf”. The screenshots for the process are shown as below.

Check each categorical variable before creating dummy variables.



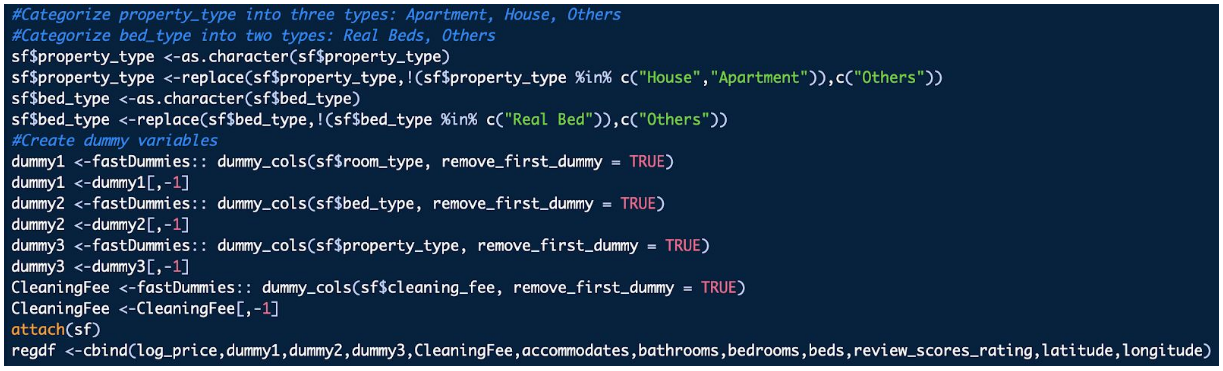
```
> table(sf$property_type)

        Apartment    Bed & Breakfast             Boat     Boutique hotel          Bungalow             Cabin
             3531                51                5                28                 8                 7
        Camper/RV   Casa particular           Castle              Cave             Chalet       Condominium
               10                 0                3                 1                 0               499
             Dorm       Earth House      Guest suite        Guesthouse            Hostel             House
               15                 0               46                28                 8              1902
              Hut            In-law           Island        Lighthouse              Loft             Other
                0                51                0                 0                85                72
    Parking Space Serviced apartment          Tent         Timeshare              Tipi         Townhouse
                0                 1                1                30                 0                46
            Train         Treehouse    Vacation home             Villa              Yurt
                0                 2                0                 2                 2
> table(sf$room_type)

Entire home/apt    Private room     Shared room
           3818            2518              98
> table(sf$bed_type)

      Airbed          Couch         Futon Pull-out Sofa      Real Bed
          39             23            54            51          6267
> table(sf$cleaning_fee)

False  True
 1642  4792
```

Abbreviation



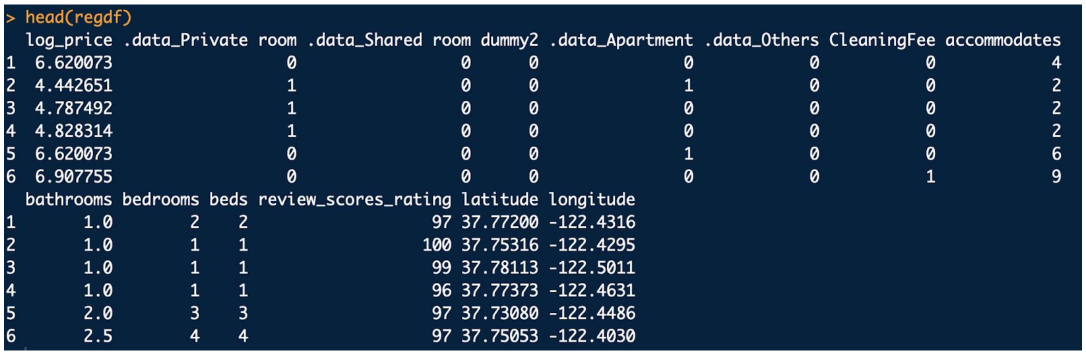
```
> sf$property_type <-as.character(sf$property_type)
> sf$property_type <-replace(sf$property_type,!(sf$property_type %in% c("House","Apartment")),c("Others"))
> sf$bed_type <-as.character(sf$bed_type)
> sf$bed_type <-replace(sf$bed_type,!(sf$bed_type %in% c("Real Bed")),c("Others"))
> table(sf$property_type)

Apartment     House    Others
     3531      1902      1001
> table(sf$bed_type)

  Others Real Bed
     167     6267
```

Create Dummy Variables



```
#Categorize property_type into three types: Apartment, House, Others
#Categorize bed_type into two types: Real Beds, Others
sf$property_type <-as.character(sf$property_type)
sf$property_type <-replace(sf$property_type,!(sf$property_type %in% c("House","Apartment")),c("Others"))
sf$bed_type <-as.character(sf$bed_type)
sf$bed_type <-replace(sf$bed_type,!(sf$bed_type %in% c("Real Bed")),c("Others"))
#Create dummy variables
dummy1 <-fastDummies:: dummy_cols(sf$room_type, remove_first_dummy = TRUE)
dummy1 <-dummy1[,-1]
dummy2 <-fastDummies:: dummy_cols(sf$bed_type, remove_first_dummy = TRUE)
dummy2 <-dummy2[,-1]
dummy3 <-fastDummies:: dummy_cols(sf$property_type, remove_first_dummy = TRUE)
dummy3 <-dummy3[,-1]
CleaningFee <-fastDummies:: dummy_cols(sf$cleaning_fee, remove_first_dummy = TRUE)
CleaningFee <-CleaningFee[,-1]
attach(sf)
regdf <-cbind(log_price,dummy1,dummy2,dummy3,CleaningFee,accommodates,bathrooms,bedrooms,beds,review_scores_rating,latitude,longitude)
```

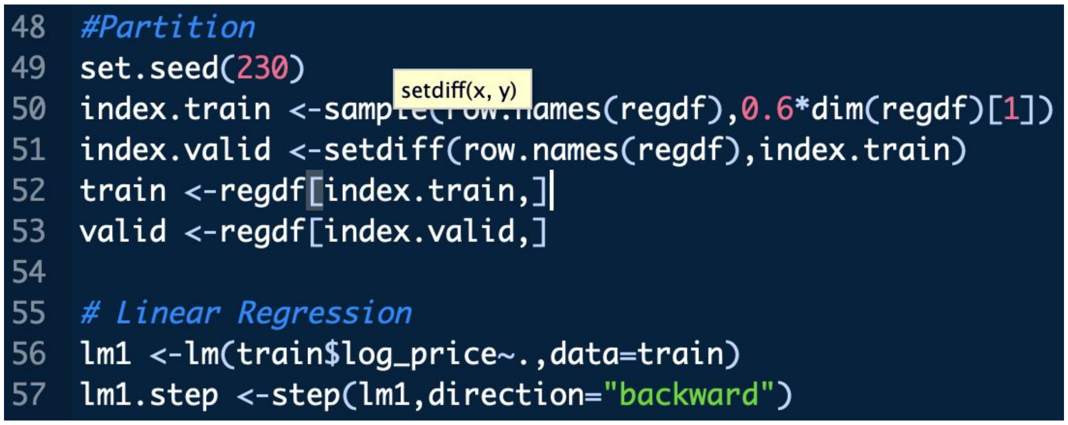
New dataframe



```
> head(regdf)
  log_price .data_Private room .data_Shared room dummy2 .data_Apartment .data_Others CleaningFee accommodates
1  6.620073                  0                 0      0               0            0           0            4
2  4.442651                  1                 0      0               1            0           0            2
3  4.787492                  1                 0      0               0            0           0            2
4  4.828314                  1                 0      0               0            0           0            2
5  6.620073                  0                 0      0               1            0           0            6
6  6.907755                  0                 0      0               0            0           1            9
  bathrooms bedrooms beds review_scores_rating latitude longitude
1       1.0        2    2                   97 37.77200 -122.4316
2       1.0        1    1                  100 37.75316 -122.4295
3       1.0        1    1                   99 37.78113 -122.5011
4       1.0        1    1                   96 37.77373 -122.4631
5       2.0        3    3                   97 37.73080 -122.4486
6       2.5        4    4                   97 37.75053 -122.4030
```

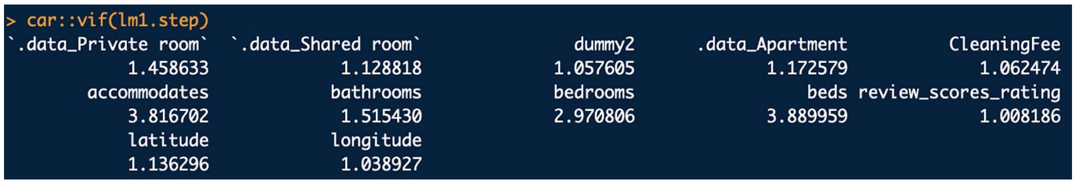
The evaluation of our model is also an important part. First, I build the multiple linear regression model based on the training data set which takes 60% of total data set, then use the summary function checking model's F-statistics, the adjusted R-square, the p-value of each coefficient. Additionally, it's also necessary to check the multicollinearity by using the vif function in the car package. As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity (James et al. 2014). After that, I'll use the predict function to predict the outcome of training data, and calculate various error terms of the difference between predicted outcomes and the log_price in the previous training data set by using the accuracy function. Repeat this step on the validating data set and then divide the RMSE by the mean of log_price in the original whole data set.

Build the model



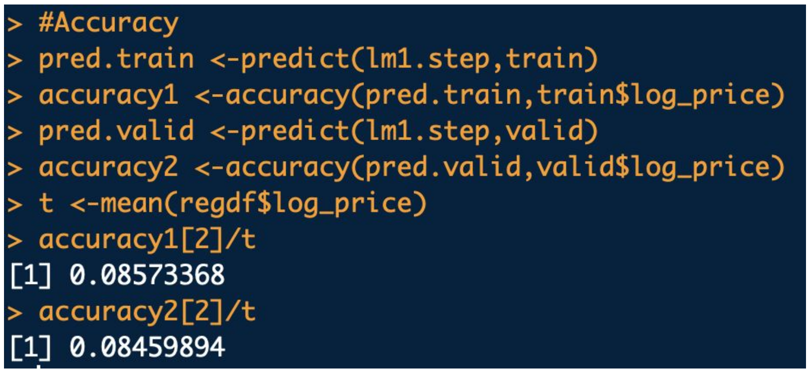
```r
#Partition
set.seed(230)
index.train <-sample(row.names(regdf),0.6*dim(regdf)[1])
index.valid <-setdiff(row.names(regdf),index.train)
train <-regdf[index.train,]
valid <-regdf[index.valid,]

# Linear Regression
lm1 <-lm(train$log_price~.,data=train)
lm1.step <-step(lm1,direction="backward")
```

Look into collinearity (All of them are lower than 5 which is good)



```
> car::vif(lm1.step)
`.data_Private room`  `.data_Shared room`               dummy2      .data_Apartment          CleaningFee
            1.458633             1.128818             1.057605             1.172579             1.062474
        accommodates            bathrooms             bedrooms                 beds review_scores_rating
            3.816702             1.515430             2.970806             3.889959             1.008186
            latitude            longitude
            1.136296             1.038927
```

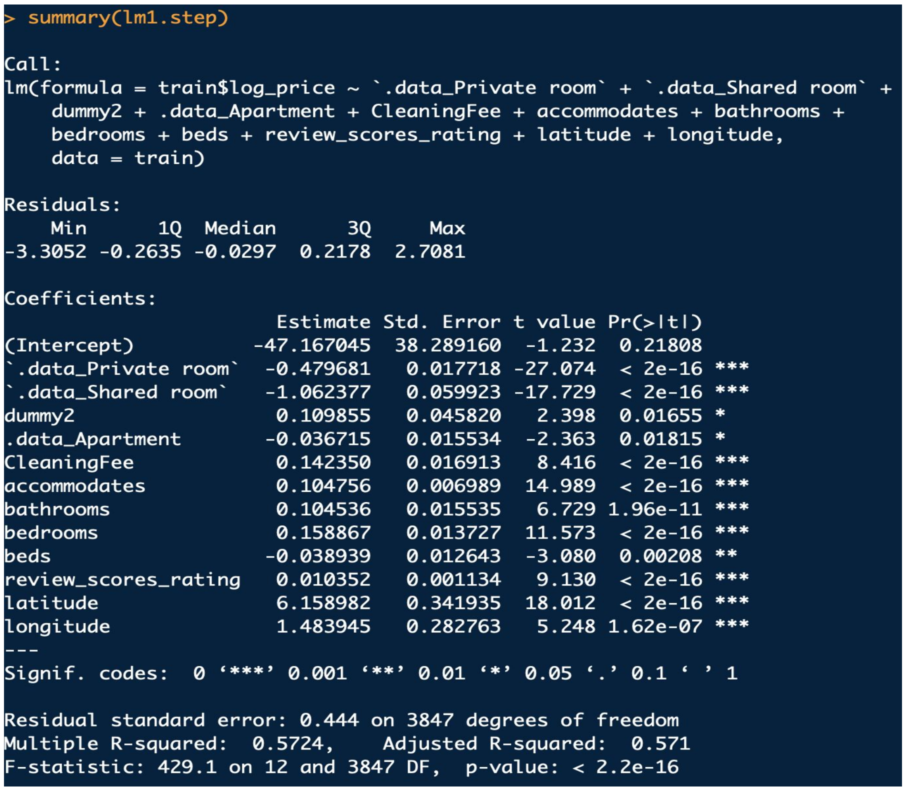
Evaluation



```
> #Accuracy
> pred.train <-predict(lm1.step,train)
> accuracy1 <-accuracy(pred.train,train$log_price)
> pred.valid <-predict(lm1.step,valid)
> accuracy2 <-accuracy(pred.valid,valid$log_price)
> t <-mean(regdf$log_price)
> accuracy1[2]/t
[1] 0.08573368
> accuracy2[2]/t
[1] 0.08459894
```

The ratios of RMSE to the mean are small enough to admit that the performance of our model is pretty good.

B) Show a screenshot of your regression summary, and explain the regression equation that it generated.

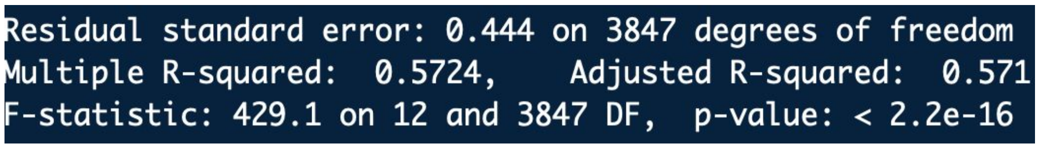


```
> summary(lm1.step)

Call:
lm(formula = train$log_price ~ `.data_Private room` + `.data_Shared room` +
    dummy2 + .data_Apartment + CleaningFee + accommodates + bathrooms +
    bedrooms + beds + review_scores_rating + latitude + longitude,
    data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3052 -0.2635 -0.0297  0.2178  2.7081

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)          -47.167045  38.289160  -1.232  0.21808
`.data_Private room`  -0.479681   0.017718 -27.074  < 2e-16 ***
`.data_Shared room`   -1.062377   0.059923 -17.729  < 2e-16 ***
dummy2                 0.109855   0.045820   2.398  0.01655 *
.data_Apartment       -0.036715   0.015534  -2.363  0.01815 *
CleaningFee            0.142350   0.016913   8.416  < 2e-16 ***
accommodates           0.104756   0.006989  14.989  < 2e-16 ***
bathrooms              0.104536   0.015535   6.729 1.96e-11 ***
bedrooms               0.158867   0.013727  11.573  < 2e-16 ***
beds                  -0.038939   0.012643  -3.080  0.00208 **
review_scores_rating   0.010352   0.001134   9.130  < 2e-16 ***
latitude               6.158982   0.341935  18.012  < 2e-16 ***
longitude              1.483945   0.282763   5.248 1.62e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.444 on 3847 degrees of freedom
Multiple R-squared:  0.5724,    Adjusted R-squared:  0.571
F-statistic: 429.1 on 12 and 3847 DF,  p-value: < 2.2e-16
```

The formula shows that the if the rental unit have private rooms, the log_price will decrease by 0.48; if the rental unit have shared rooms, the log_price will decrease by 1.06; if the rental unit does not have real beds, the log_price will increase by 0.11; if the rental unit is an apartment, the log_price will decrease by 0.03; if the rental unit charges cleaning fee, the log_price will increase by 0.14; if the rental unit can accommodate additional person, the log_price will increase by 0.10; if the rental unit have an additional bathroom, the log_price will increase by 0.10; if the rental unit have an additional bedroom, the log_price will by 0.16; if the rental unit have an additional bed, the log_price will go down by 0.04; if review score of the rental unit increases by one unit, the log_price will increase by 0.01; if the rental unit has one unit increase on latitude, the log_price will increase by 6.16; if the rental unit has one unit increase on longitude, the log_price will increase by 1.48.

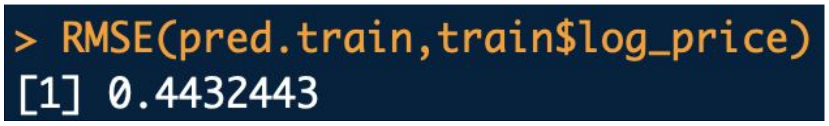
a) **What is the r-squared of your model? What does this mean?**

Due to the fact that it is a multiple regression model, we need to focus on the adjusted R-squared rather than Multiple R-squared. The adjusted R-squared is 0.571. It means that about 57.1% of the independable variable can be explained by the model's dependable variables. 57.1% is a relatively good threshold that means the independable variables can fit the regression line of model well.

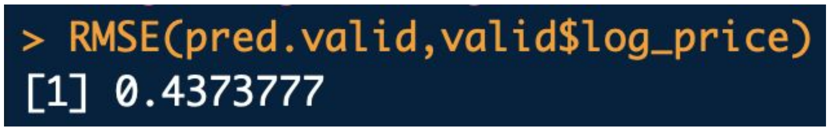


```
Residual standard error: 0.444 on 3847 degrees of freedom
Multiple R-squared:  0.5724,    Adjusted R-squared:  0.571
F-statistic: 429.1 on 12 and 3847 DF,  p-value: < 2.2e-16
```

b) What is the RMSE of your model? What does this mean?



```
> RMSE(pred.train,train$log_price)
[1] 0.4432443
```



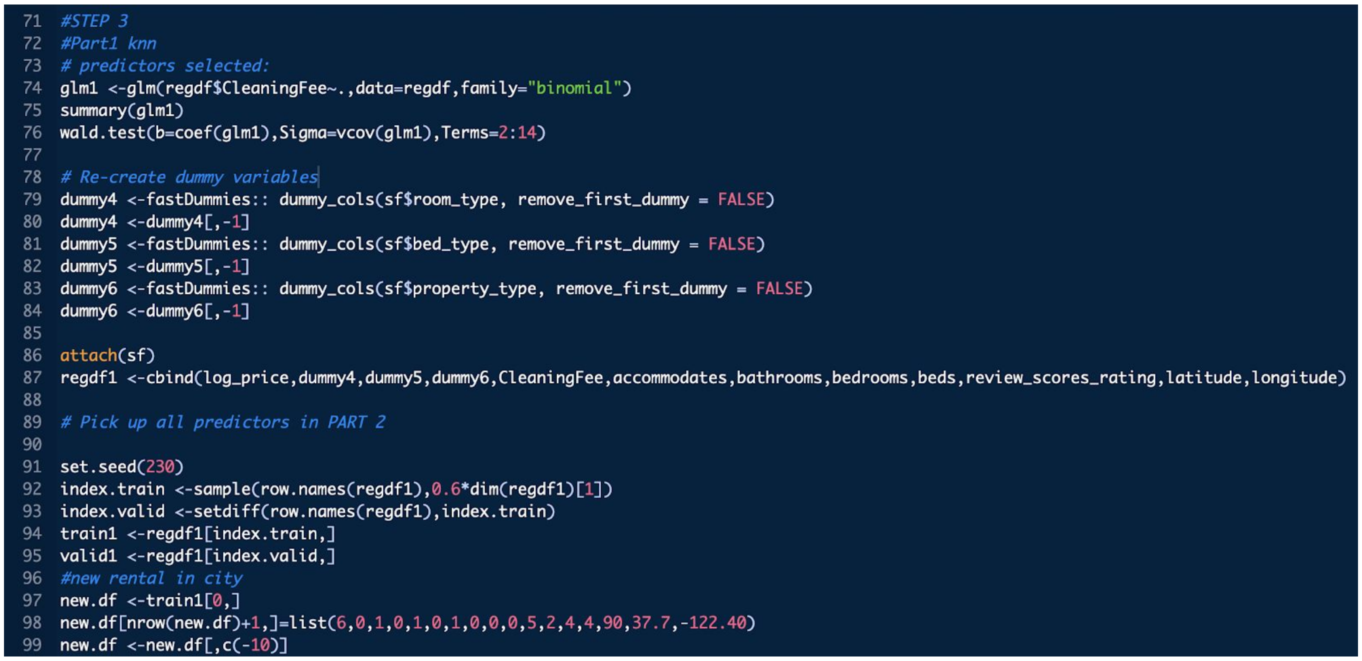
```
> RMSE(pred.valid,valid$log_price)
[1] 0.4373777
```

The RMSE of training data set model is rounded 0.44, and that of my validation data set model is rounded 0.44 as well. RMSE measures the distribution of model residuals. The higher the RMSE, the more discrete the distribution. The range is 0 to 1. However, there is no an exact threshold to evaluate whether it is good or bad, because it depends on the scale on dependable variable. We need to normalize it. We are supposed to divide the RMSE by the mean of log_price. Here, we can get the normalized RMSE is 0.086, which is extremely small. In addition, the closer the RMSE for training set and validation sets, the better fit of the model. Therefore, the RMSE of our model are both similar and good.

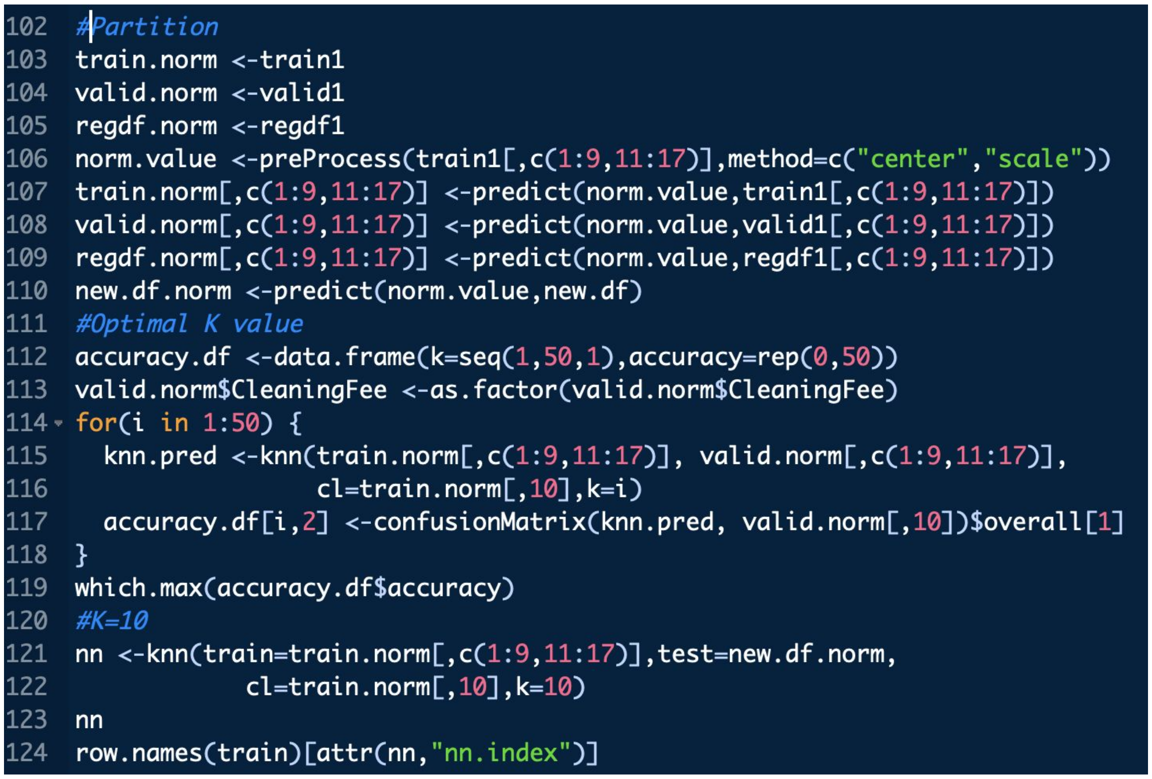
Step III: Classification (40 points)

<u>Part I. Using k-nearest neighbors</u>, predict whether a rental in your city would have a cleaning fee. Use any set of predictors in order to build this model.

A. Show the code you used to run your model, and the code you used to assess your model.



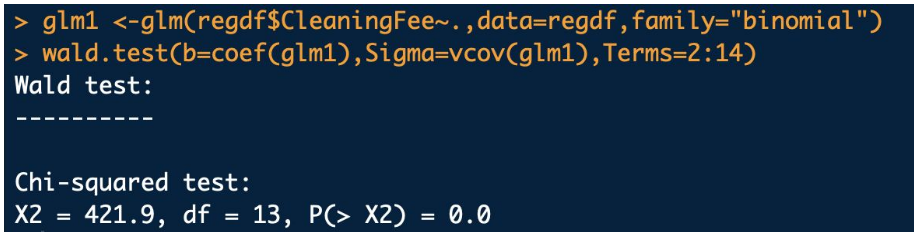
```r
#STEP 3
#Part1 knn
# predictors selected:
glm1 <-glm(regdf$CleaningFee~.,data=regdf,family="binomial")
summary(glm1)
wald.test(b=coef(glm1),Sigma=vcov(glm1),Terms=2:14)

# Re-create dummy variables
dummy4 <-fastDummies:: dummy_cols(sf$room_type, remove_first_dummy = FALSE)
dummy4 <-dummy4[,-1]
dummy5 <-fastDummies:: dummy_cols(sf$bed_type, remove_first_dummy = FALSE)
dummy5 <-dummy5[,-1]
dummy6 <-fastDummies:: dummy_cols(sf$property_type, remove_first_dummy = FALSE)
dummy6 <-dummy6[,-1]

attach(sf)
regdf1 <-cbind(log_price,dummy4,dummy5,dummy6,CleaningFee,accommodates,bathrooms,bedrooms,beds,review_scores_rating,latitude,longitude)

# Pick up all predictors in PART 2

set.seed(230)
index.train <-sample(row.names(regdf1),0.6*dim(regdf1)[1])
index.valid <-setdiff(row.names(regdf1),index.train)
train1 <-regdf1[index.train,]
valid1 <-regdf1[index.valid,]
#new rental in city
new.df <-train1[0,]
new.df[nrow(new.df)+1,]=list(6,0,1,0,1,0,1,0,0,0,5,2,4,4,90,37.7,-122.40)
new.df <-new.df[,c(-10)]
```



```r
#Partition
train.norm <-train1
valid.norm <-valid1
regdf.norm <-regdf1
norm.value <-preProcess(train1[,c(1:9,11:17)],method=c("center","scale"))
train.norm[,c(1:9,11:17)] <-predict(norm.value,train1[,c(1:9,11:17)])
valid.norm[,c(1:9,11:17)] <-predict(norm.value,valid1[,c(1:9,11:17)])
regdf.norm[,c(1:9,11:17)] <-predict(norm.value,regdf1[,c(1:9,11:17)])
new.df.norm <-predict(norm.value,new.df)
#Optimal K value
accuracy.df <-data.frame(k=seq(1,50,1),accuracy=rep(0,50))
valid.norm$CleaningFee <-as.factor(valid.norm$CleaningFee)
for(i in 1:50) {
  knn.pred <-knn(train.norm[,c(1:9,11:17)], valid.norm[,c(1:9,11:17)],
                 cl=train.norm[,10],k=i)
  accuracy.df[i,2] <-confusionMatrix(knn.pred, valid.norm[,10])$overall[1]
}
which.max(accuracy.df$accuracy)
#K=10
nn <-knn(train=train.norm[,c(1:9,11:17)],test=new.df.norm,
              cl=train.norm[,10],k=10)
nn
row.names(train)[attr(nn,"nn.index")]
```

B. Write a two-paragraph narrative that describes how you did this. In your narrative, be sure to describe your predictor choices, and mention how you arrived at the particular k value that you used.

To begin with, I need to determine predictors that I will put into my model. The MLR model built at prediction part can be regarded as accurate so that I want to use those parameters as my predictors in this model too. The tiny change is that the variable Log_price is exchanged with Cleaning Fee. Nonetheless, the dependable variable at this part is Cleaning Fee with two possible values, yes or no. Therefore, we should build a logistic regression model by using glm() function in R and then test all parameters. If the new model can pass the Wald-test at alpha=0.05 levels, which means the p-value is smaller than 0.05, then it could verifies that parameters in the new model are significant.



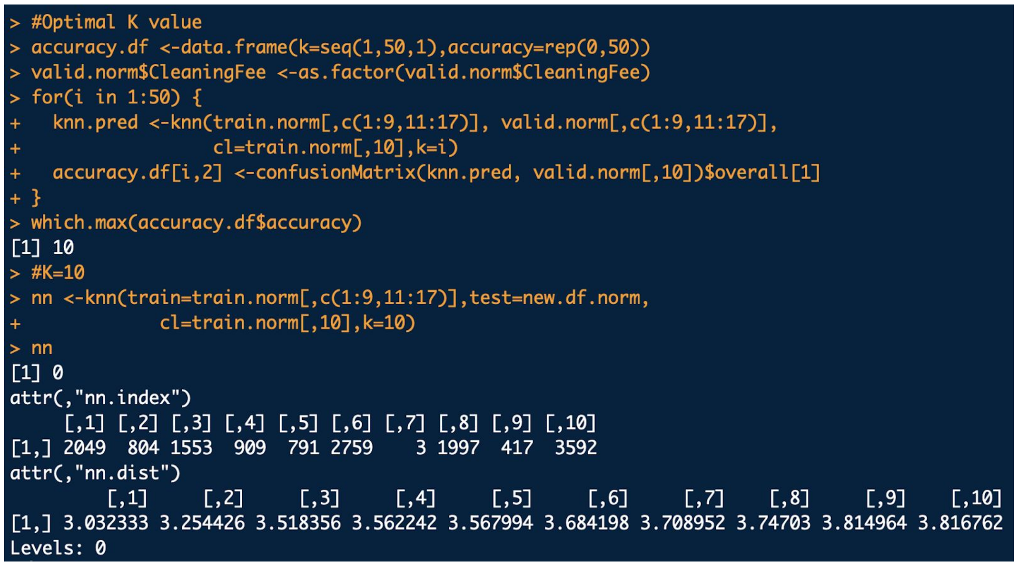
```
> glm1 <-glm(regdf$CleaningFee~.,data=regdf,family="binomial")
> wald.test(b=coef(glm1),Sigma=vcov(glm1),Terms=2:14)
Wald test:
----------

Chi-squared test:
X2 = 421.9, df = 13, P(> X2) = 0.0
```

We can see that the p-value is 0, less than 0.05, so those parameters are significant to the dependable variable Cleaning Fee.

Another thing is that in the prediction part, we have to avoid perfect-collinearity so we remove the first dummy variable for each parameter. But for KNN, we don't need to avoid perfect-collinearity because KNN measures distance among individual points. Thus, I recreate the dummy variables without removing the first dummy.

Next, I create a new dataframe and input explanatory variables for simulating the new rental in San Francisco. The data partition part is same as the Part II, 60% in training set and 40% in validation set. For KNN, the data must be normalized so that distance can be calculated with no bias. So I use the preProcess function to normalize the training set, then use the predict function to normalize the rested data sets.

For determining the optimal K-value, I should build up a KNN prediction model first. The Cleaning Fee in the normalized training set is treated as the classification factor. Then I calculate the accuracy for validation set and KNN prediction model by picking up different K-value 49 times from 2 to 50. The highest accuracy accords to k-value that equals to 10. The final step is to use the knn function again, but this time it will generate the level where the self-setted rental unit falls into.

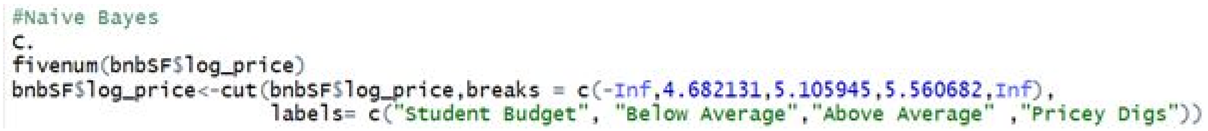


```
> #Optimal K value
> accuracy.df <-data.frame(k=seq(1,50,1),accuracy=rep(0,50))
> valid.norm$CleaningFee <-as.factor(valid.norm$CleaningFee)
> for(i in 1:50) {
+   knn.pred <-knn(train.norm[,c(1:9,11:17)], valid.norm[,c(1:9,11:17)],
+                  cl=train.norm[,10],k=i)
+   accuracy.df[i,2] <-confusionMatrix(knn.pred, valid.norm[,10])$overall[1]
+ }
> which.max(accuracy.df$accuracy)
[1] 10
> #K=10
> nn <-knn(train=train.norm[,c(1:9,11:17)],test=new.df.norm,
+              cl=train.norm[,10],k=10)
> nn
[1] 0
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] 2049  804 1553  909  791 2759    3 1997  417  3592
attr(,"nn.dist")
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]    [,8]     [,9]    [,10]
[1,] 3.032333 3.254426 3.518356 3.562242 3.567994 3.684198 3.708952 3.74703 3.814964 3.816762
Levels: 0
```

The results show that my new rental unit will fall into level 0 which means there is no cleaning fee charged. It also shows that my 10 nearest neighbors are row 2049, 804, 1553, 909, 791, 3, 1997, 417 and 3592. The last row in the screenshot means their measured distances.

Part II. Naive Bayes

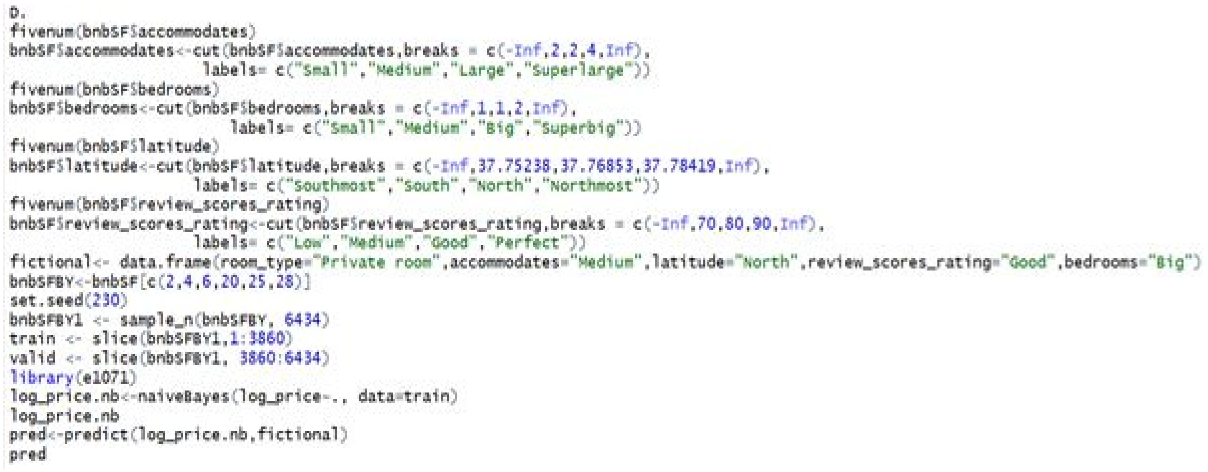
C. Using the log_price variable, create four similarly-sized bins, or categories, for the rental prices in your city (Pricey Digs, Above Average, Below Average, Student Budget).



```
#Naive Bayes
C.
fivenum(bnbSF$log_price)
bnbSF$log_price<-cut(bnbSF$log_price,breaks = c(-Inf,4.682131,5.105945,5.560682,Inf),
                     labels= c("Student Budget", "Below Average","Above Average" ,"Pricey Digs"))
```

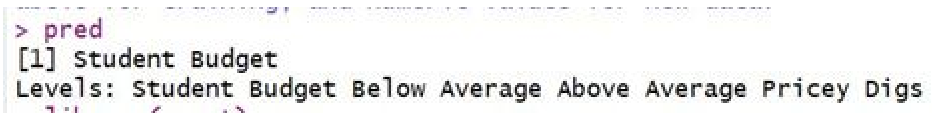
In this step, we were using log_price as our variable, and created four similarly-sized categories for the rental price in San Francisco. First, we use >fivenum(bnbSF$log_price) to find the 1/4 quartile, mean, and 3/4 quartile for the rental price. Then we use these three numbers to cut the rental price into four ranges > c(-Inf,4.682131,5.105945,5.560682,Inf) and labeled them as ("Student Budget", "Below Average", "Above Average" ,"Pricey Digs").

D. Using any five predictors, build a model using the naive Bayes algorithm. Describe a fictional apartment, and use your model to predict which bin it will fall into.



```
D.
fivenum(bnbSF$accommodates)
bnbSF$accommodates<-cut(bnbSF$accommodates,breaks = c(-Inf,2,2,4,Inf),
                     labels= c("Small","Medium","Large","Superlarge"))
fivenum(bnbSF$bedrooms)
bnbSF$bedrooms<-cut(bnbSF$bedrooms,breaks = c(-Inf,1,1,2,Inf),
                     labels= c("Small","Medium","Big","Superbig"))
fivenum(bnbSF$latitude)
bnbSF$latitude<-cut(bnbSF$latitude,breaks = c(-Inf,37.75238,37.76853,37.78419,Inf),
                     labels= c("Southmost","South","North","Northmost"))
fivenum(bnbSF$review_scores_rating)
bnbSF$review_scores_rating<-cut(bnbSF$review_scores_rating,breaks = c(-Inf,70,80,90,Inf),
                     labels= c("Low","Medium","Good","Perfect"))
fictional<- data.frame(room_type="Private room",accommodates="Medium",latitude="North",review_scores_rating="Good",bedrooms="Big")
bnbSFBY<-bnbSF[c(2,4,6,20,25,28)]
set.seed(230)
bnbSFBY1 <- sample_n(bnbSFBY, 6434)
train <- slice(bnbSFBY1,1:3860)
valid <- slice(bnbSFBY1, 3860:6434)
library(e1071)
log_price.nb<-naiveBayes(log_price~., data=train)
log_price.nb
pred<-predict(log_price.nb,fictional)
pred
```

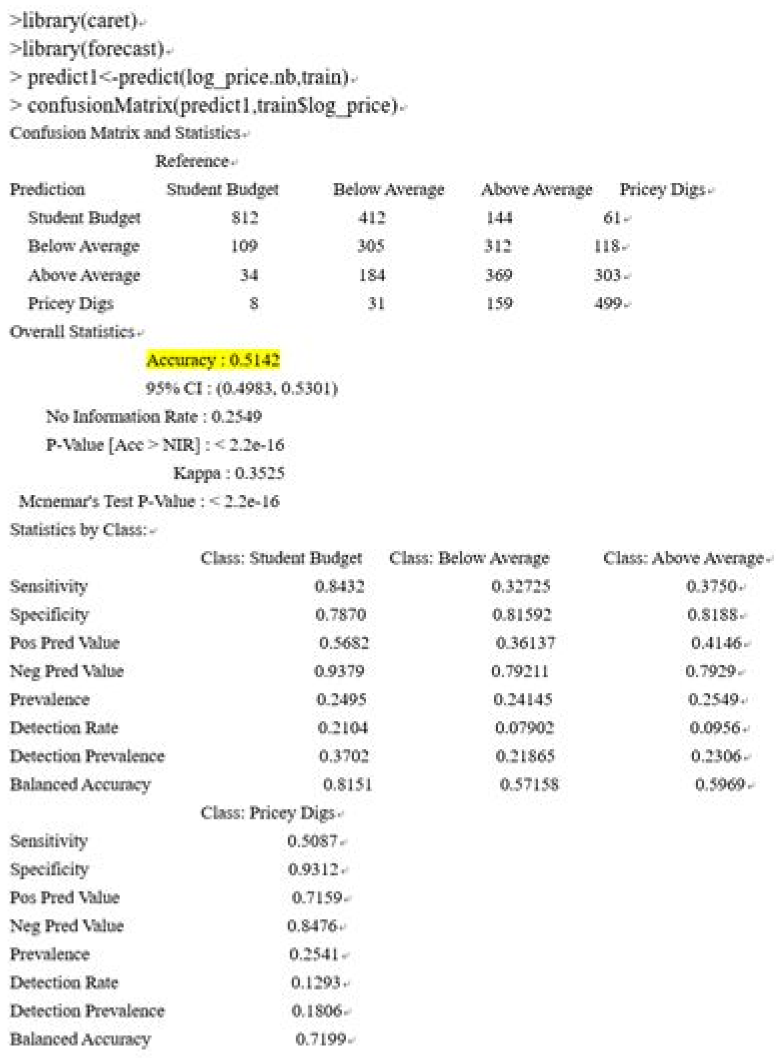
Based on the regression we ran in Step II, we found the top five predictors that are most related to log_price were room_type, accommodates, latitude, review_scores_rating, and bedrooms. So, we decided to use these five predictors to build a model. Firstly, we changed accommodates, bedrooms, latitude, and review_scores_rating to categorical variables. By using the “fivenum” function, we made each of these four variables into four categories. We cut the accommodates number into four parts and labeled as (“Small”, “Medium”, “Large”, “Super Large”). We cut bedrooms number into four parts and labeled as (“Small”, “Medium”, “Big”, “Superbig”). We cut latitude into four parts and labeled as (“Southmost”, “South”, “North”, “Northmost”). We cut review_scores_rating in to four parts and labeled as (“Low”, “Medium”, “Good”, “Perfect”).



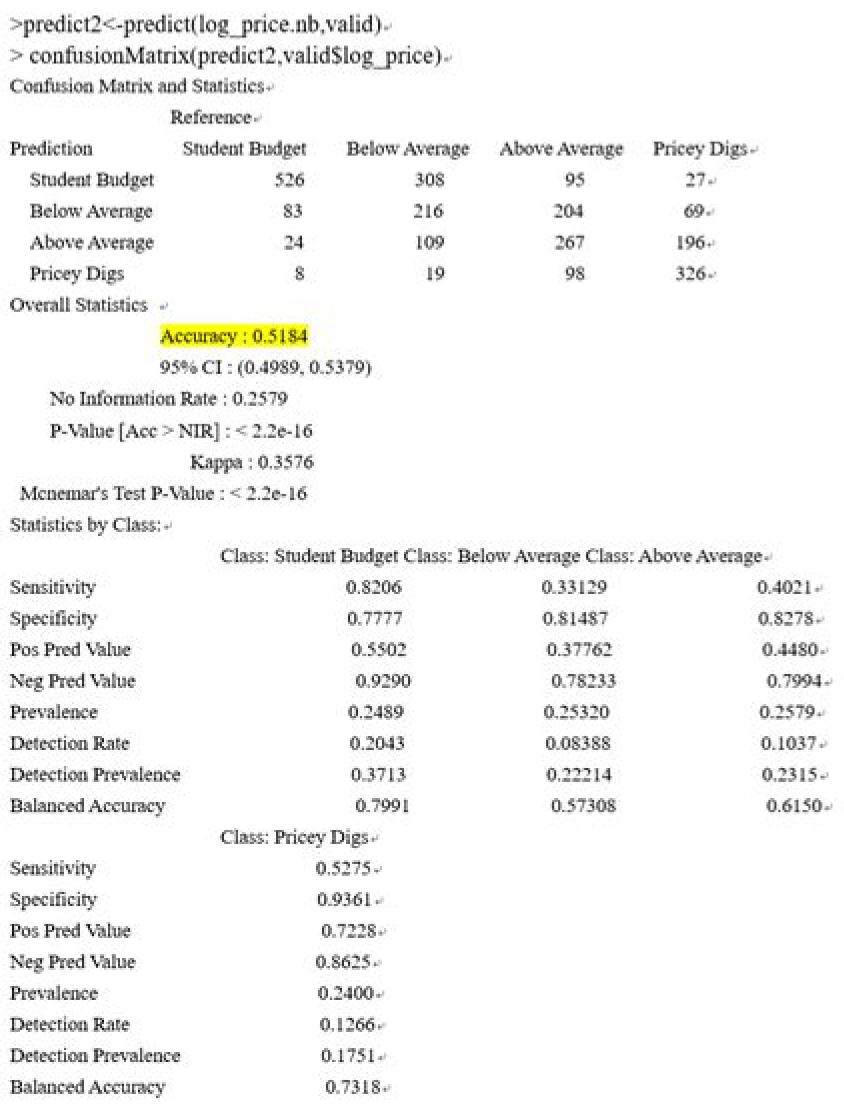
```
> pred
[1] Student Budget
Levels: Student Budget Below Average Above Average Pricey Digs
```

We created a fictional apartment, which room_type is “Private room”, accommodates equals “Medium”, bedrooms is “Big”, with “North” latitude and “Good” review_scores_rating. By using our model to do the prediction, our fictional apartment fell into “Student Budget”. That means the log_price of our fictional apartment is low.

E. Show a screenshot of the code you used to build your model, the code you used to run the algorithm, and code you used to assess the algorithm.



```
>library(caret)
>library(forecast)
> predict1<-predict(log_price.nb,train)
> confusionMatrix(predict1,train$log_price)
Confusion Matrix and Statistics
                  Reference
Prediction        Student Budget    Below Average    Above Average    Pricey Digs
   Student Budget          812              412              144             61
   Below Average           109              305              312            118
   Above Average            34              184              369            303
   Pricey Digs               8               31              159            499
Overall Statistics
                 Accuracy : 0.5142
                   95% CI : (0.4983, 0.5301)
     No Information Rate : 0.2549
     P-Value [Acc > NIR] : < 2.2e-16
                    Kappa : 0.3525
  Mcnemar's Test P-Value : < 2.2e-16
Statistics by Class:
                      Class: Student Budget  Class: Below Average  Class: Above Average
Sensitivity                          0.8432               0.32725                0.3750
Specificity                          0.7870               0.81592                0.8188
Pos Pred Value                       0.5682               0.36137                0.4146
Neg Pred Value                       0.9379               0.79211                0.7929
Prevalence                           0.2495               0.24145                0.2549
Detection Rate                       0.2104               0.07902                0.0956
Detection Prevalence                 0.3702               0.21865                0.2306
Balanced Accuracy                    0.8151               0.57158                0.5969
                      Class: Pricey Digs
Sensitivity                      0.5087
Specificity                      0.9312
Pos Pred Value                   0.7159
Neg Pred Value                   0.8476
Prevalence                       0.2541
Detection Rate                   0.1293
Detection Prevalence             0.1806
Balanced Accuracy                0.7199
```



```
>predict2<-predict(log_price.nb,valid)
> confusionMatrix(predict2,valid$log_price)
Confusion Matrix and Statistics
                    Reference
Prediction          Student Budget    Below Average    Above Average    Pricey Digs
   Student Budget              526              308               95             27
   Below Average                83              216              204             69
   Above Average                24              109              267            196
   Pricey Digs                   8               19               98            326
Overall Statistics
                   Accuracy : 0.5184
                   95% CI : (0.4989, 0.5379)
     No Information Rate : 0.2579
     P-Value [Acc > NIR] : < 2.2e-16
                      Kappa : 0.3576
 Mcnemar's Test P-Value : < 2.2e-16
Statistics by Class:
                          Class: Student Budget Class: Below Average Class: Above Average
Sensitivity                           0.8206             0.33129               0.4021
Specificity                           0.7777             0.81487               0.8278
Pos Pred Value                        0.5502             0.37762               0.4480
Neg Pred Value                        0.9290             0.78233               0.7994
Prevalence                            0.2489             0.25320               0.2579
Detection Rate                        0.2043             0.08388               0.1037
Detection Prevalence                  0.3713             0.22214               0.2315
Balanced Accuracy                     0.7991             0.57308               0.6150
                          Class: Pricey Digs
Sensitivity                   0.5275
Specificity                   0.9361
Pos Pred Value                0.7228
Neg Pred Value                0.8625
Prevalence                    0.2400
Detection Rate                0.1266
Detection Prevalence          0.1751
Balanced Accuracy             0.7318
```

F. Write a two-paragraph narrative that describes how you did this. In your narrative, be sure to talk about things like factor selection and testing against your training data.

First, we cut the log_price into four bins and chose five most related predictors to build our Naïve Bayes model. We created a fictional apartment, which room_type is “Private room”, accommodates equals “Medium”, bedrooms is “Big”, with “North” latitude and “Good” review_scores_rating. Based on our prediction, our fictional apartment fell into “Student Budget”.

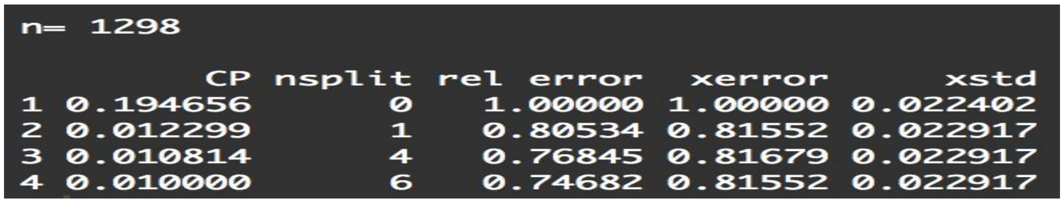
We want to see how accuracy is this result. So, we set seed as 230 and sliced our data into a training data set (1:3860) and a validation data set (3860:6434). “Predict” function and “confusionMatrix” function were using to get the accuracy. We got 0.5142 for the accuracy of training data set and 0.5184 for the accuracy of validation data set.

These two accuracy numbers are quite same, that means our model is performing well. However, both accuracies are low. It might because there are many predictors in the model, but we just chose five predictors that most related to log_price. Different groups of people have different key factors when they choose apartment. So, it is hard to say most people are using these five factors to measure the log_price. That might be the reason why the accuracy is low.

Part III: Classification Tree

A. Build a classification tree that predicts the cancellation policy of an airbnb rental listing, with the outcome classes “flexible”, “moderate”, and “strict.”

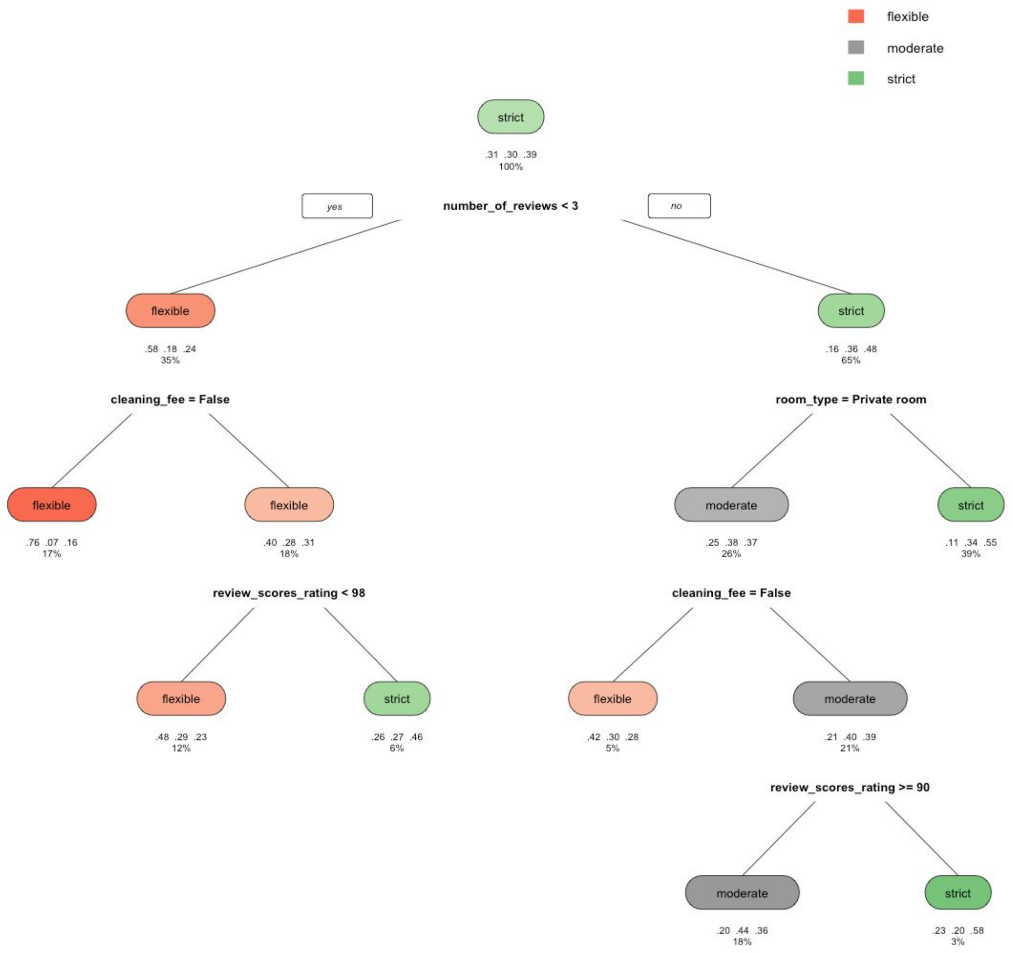
B. Determine the ideal size of your tree using cross-validation.



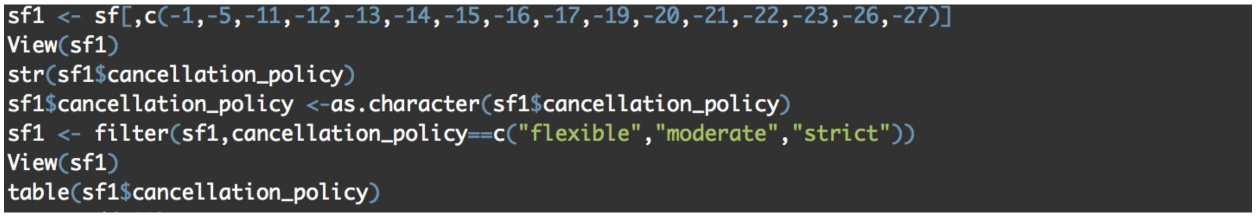
```
n= 1298

          CP nsplit rel error  xerror     xstd
1 0.194656      0   1.00000 1.00000 0.022402
2 0.012299      1   0.80534 0.81552 0.022917
3 0.010814      4   0.76845 0.81679 0.022917
4 0.010000      6   0.74682 0.81552 0.022917
```

According to the listed data, the ideal size of our tree will be 6 since it was with the smallest error and xerror.

C. Using rpart.plot and your choice of graphical parameters, show your tree model here.

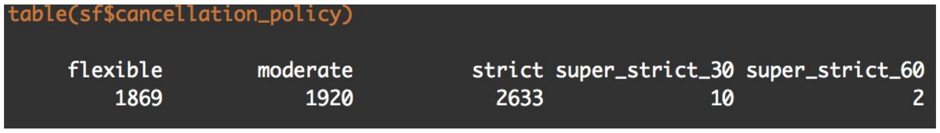


D. In a 1-2 paragraph write-up, describe your process. Talk about some of the features that you considered using, and your reasons why. Mention anything that you found interesting as you explored various possible models, and the process you used to arrive at the model you finished with.

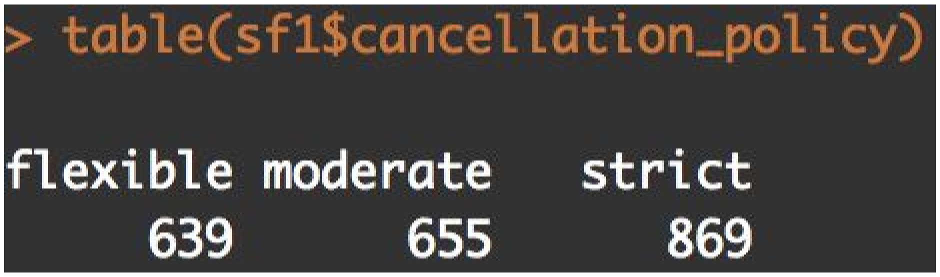


```
sf1 <- sf[,c(-1,-5,-11,-12,-13,-14,-15,-16,-17,-19,-20,-21,-22,-23,-26,-27)]
View(sf1)
str(sf1$cancellation_policy)
sf1$cancellation_policy <-as.character(sf1$cancellation_policy)
sf1 <- filter(sf1,cancellation_policy==c("flexible","moderate","strict"))
View(sf1)
table(sf1$cancellation_policy)
```

At the first in our model, I decide the "number of reviews", "review scores rating", "room type", and "cleaning fee" as our input predictors. These predictors reflect what the overall quality and performance of rent look like. I remove a few variables which would be useless for me to build the tree model. The remaining variables in the new data frame (sf1) include the necessary variables, such as log price, room type, number of reviews, review scores, relating to the cancellation policy, which is regarded as the necessary variables in our model so that they are the closest factors influencing the cancellation policy. In the dataset, I've realized that the cancellation policy has 5 levels: flexible, moderate, strict, strict 30, and strict 60. Since the last two factors have the smallest amount, I then set these two factors to 0 and drop them. After setting up the new data frame sf1, then the number of the remaining factors are listed as follows: 639 for flexible, 655 for moderate, and 869 for strict.

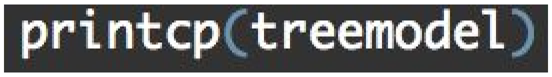


```
table(sf$cancellation_policy)

    flexible        moderate          strict super_strict_30 super_strict_60
        1869            1920            2633              10               2
```



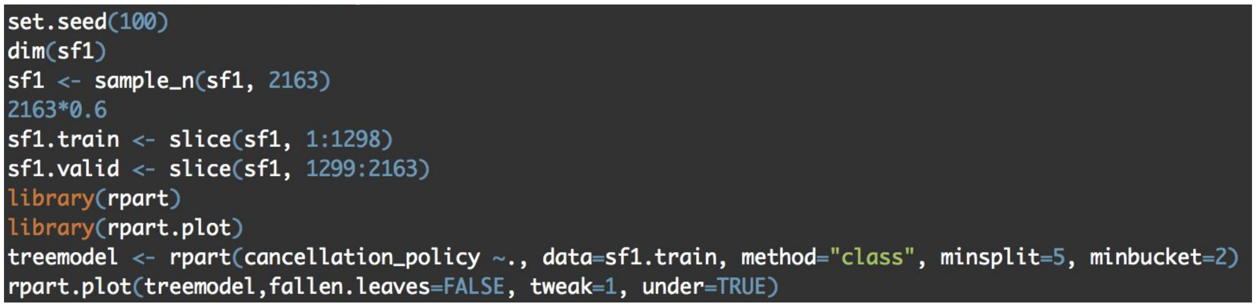
```
> table(sf1$cancellation_policy)

flexible moderate   strict
     639      655      869
```

Then I use the cross-validation method, applying printcp function to validate the tree and to find the ideal size of our model. Refer to question B in this part, it is obviously shown that the ideal size for our model is 6 as it has the smallest rel-error and xerror. The reason I prune the tree is to avoid any overfitting of the data, in the purpose of getting a small tree and the one with least cross-validated error.

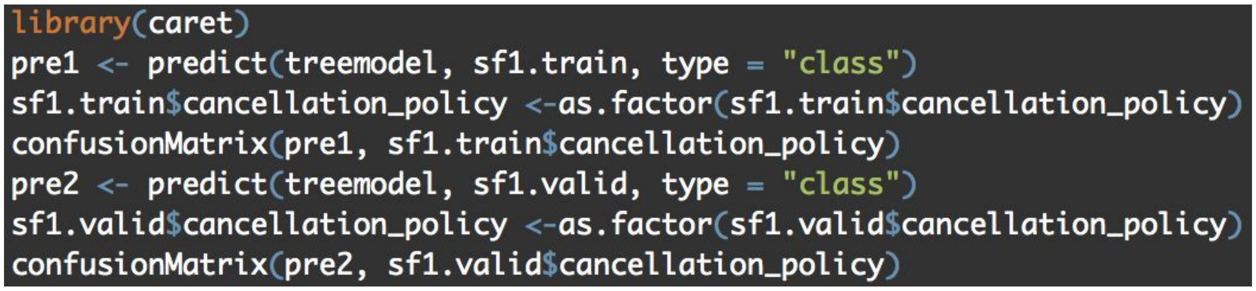


```
printcp(treemodel)
```

Later than, I prepared to set up training and validation dataset to test the accuracy of our tree model, then applied the rpart function to build the model, along with the minsplit and minbucket to show the different outlook of the model. Function rpart.plot is utilized to show the big picture of the model tree. Our tree model does identify the classification of rules. To interpret the tree model, I can explain on the basis of one situation. Let's say, from the observation of 100% strict circumstance, if the answer of variables "number of reviews < 3" is no, there will be 65% of strict type with room type = private room. And from this observation, 39% of them can be classified as strict.



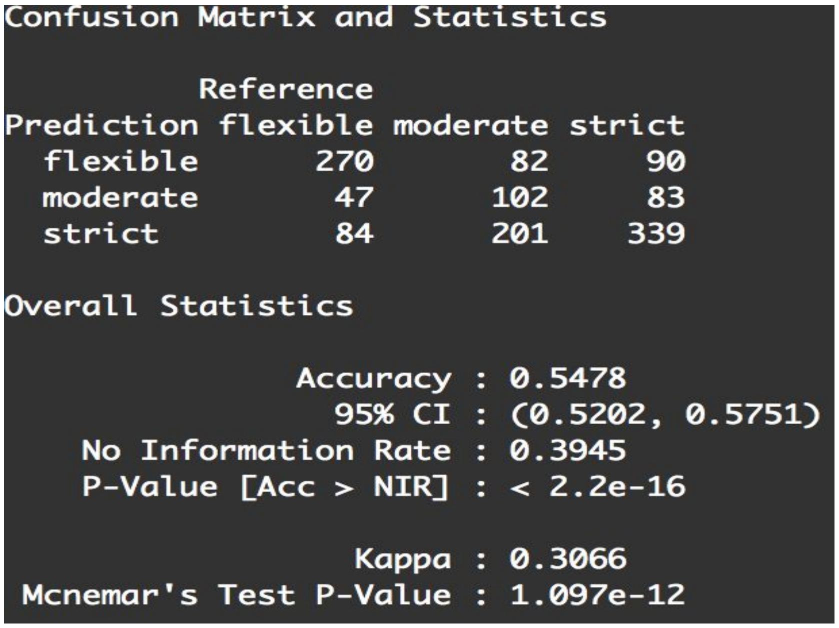
```
set.seed(100)
dim(sf1)
sf1 <- sample_n(sf1, 2163)
2163*0.6
sf1.train <- slice(sf1, 1:1298)
sf1.valid <- slice(sf1, 1299:2163)
library(rpart)
library(rpart.plot)
treemodel <- rpart(cancellation_policy ~., data=sf1.train, method="class", minsplit=5, minbucket=2)
rpart.plot(treemodel,fallen.leaves=FALSE, tweak=1, under=TRUE)
```



```
library(caret)
pre1 <- predict(treemodel, sf1.train, type = "class")
sf1.train$cancellation_policy <-as.factor(sf1.train$cancellation_policy)
confusionMatrix(pre1, sf1.train$cancellation_policy)
pre2 <- predict(treemodel, sf1.valid, type = "class")
sf1.valid$cancellation_policy <-as.factor(sf1.valid$cancellation_policy)
confusionMatrix(pre2, sf1.valid$cancellation_policy)
```

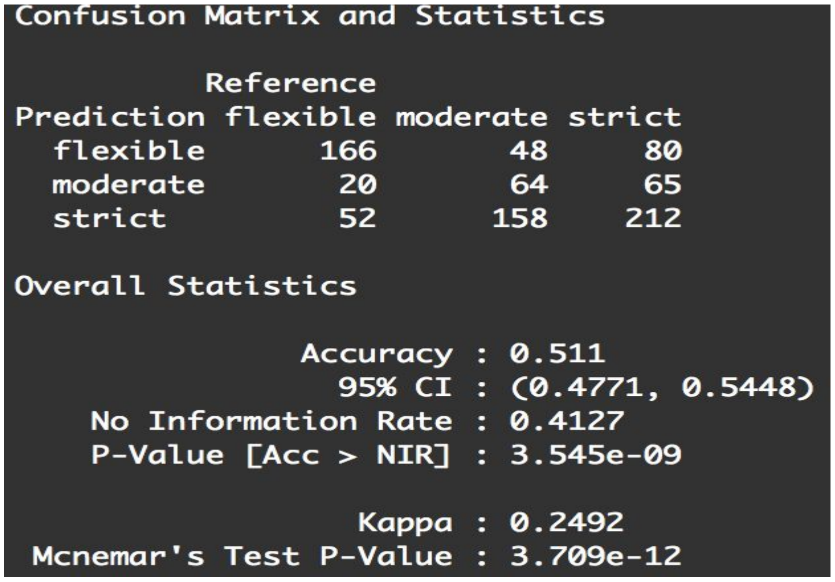
The last step I've done is to predict the accuracy of my training and validation dataset. Even though from common sense we have known the training set has the most accurate, this process is necessary to double check in order to get the most accurate model with reliable data. Refer to the information below, with no doubt the results show that the accuracy of training dataset has a higher number of 0.5478 than the validation has (0.511).

Train:



```
Confusion Matrix and Statistics

          Reference
Prediction flexible moderate strict
  flexible      270       82     90
  moderate       47      102     83
  strict         84      201    339

Overall Statistics

               Accuracy : 0.5478
                 95% CI : (0.5202, 0.5751)
    No Information Rate : 0.3945
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3066
 Mcnemar's Test P-Value : 1.097e-12
```

Valid:



```
Confusion Matrix and Statistics

          Reference
Prediction flexible moderate strict
  flexible      166       48     80
  moderate       20       64     65
  strict         52      158    212

Overall Statistics

               Accuracy : 0.511
                 95% CI : (0.4771, 0.5448)
    No Information Rate : 0.4127
    P-Value [Acc > NIR] : 3.545e-09

                  Kappa : 0.2492
 Mcnemar's Test P-Value : 3.709e-12
```

Meanwhile, I found the interesting thing when I prepared to build up the tree model. We can recognize that almost decision nodes of our tree model are classified with the variables “review scores rating”. From this, I’d like to understand this as a reasonable situation since the scores rating plays a vital role for customers to make the decision of choosing a booking which has a strict cancellation policy.

Step IV: Clustering (15 points)
I. Perform either a k-means analysis or a hierarchical clustering analysis to compare any selected list of neighborhoods in your city.

**** You may find this task far more manageable if you reduce the number of neighborhoods that you're dealing with. The results will be more meaningful that way, too.**

**** Of any section of the project, this one offers the most opportunity to be creative and take risks. Think about feature engineering, too -- how/when/where can you create new variables based on existing ones?**

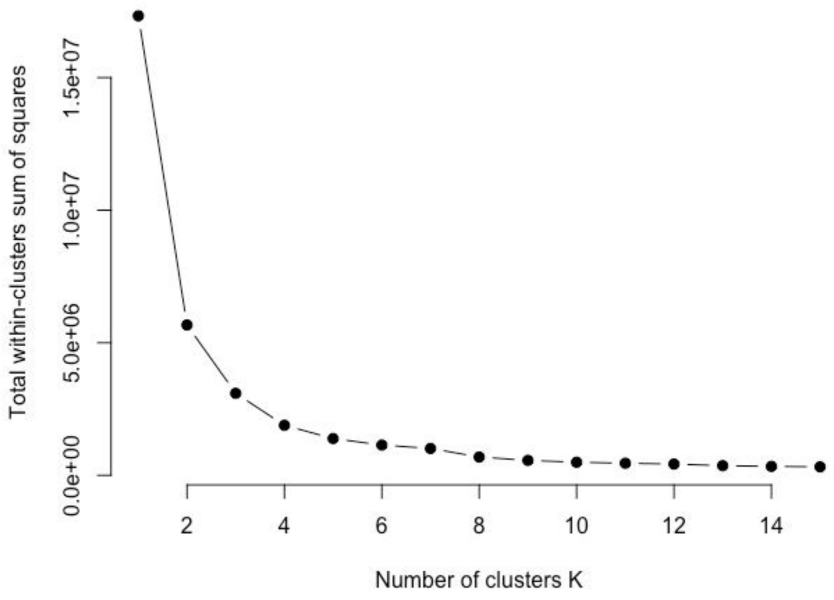
**** group_by() and summarize() from the tidyverse can be quite helpful here (or, if you prefer to use aggregate() and tapply(), those can help to accomplish a similar task.**

II. Show your code and results, and write two paragraphs describing what you included in your model, and what you found about the neighborhoods. You do not need to bring in any outside sources here to assess your findings. (in other words, you don't need to actually go out and research the neighborhoods).
Step IV: Clustering

```
df <- read.csv("airbnbtrain.csv")
sf <- subset(df,city=="SF")
sf$city <- droplevels(sf$city)

sf$bathrooms[is.na(sf$bathrooms)] <- median(sf$bathrooms,na.rm=TRUE)
sf$bedrooms[is.na(sf$bedrooms)] <- median(sf$bedrooms,na.rm=TRUE)
sf$beds[is.na(sf$beds)] <- median(sf$beds,na.rm=TRUE)
sf$review_scores_rating[is.na(sf$review_scores_rating)] <-
median(sf$review_scores_rating,na.rm=TRUE)

sfcluster <- sf[, -c(1,3,4,5,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,26,27)]
View(sfcluster)
```

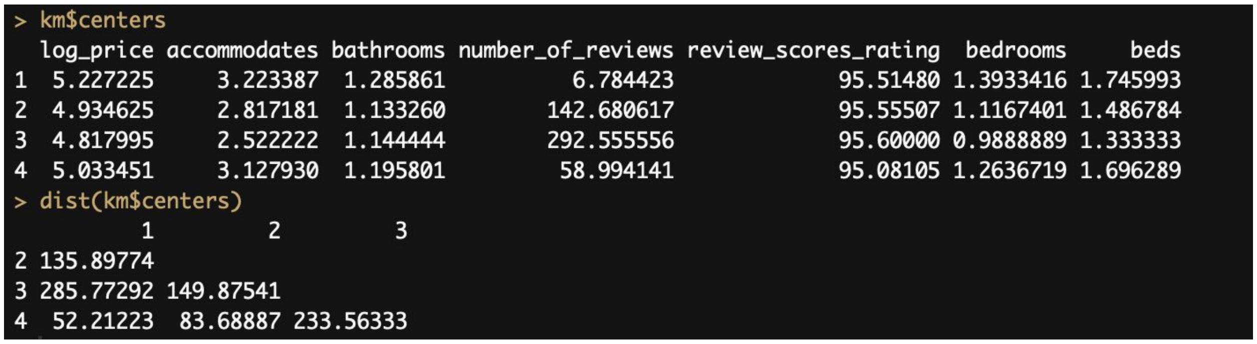
In the beginning, data should be prepared for better clustering analysis for only columns which are considered as useful from SF dataset would be kept in building clusters. Same as the process in Step I, missing values from “bathrooms”, “bedrooms”, “beds”, and “review_score_rating” are replaced with their median values respectively. Then, useless columns are deleted from the dataset, keeping potential influential factors, including price, accommodates, number of reviews, bedrooms, and beds, as well as review score ratings. Dummy variables are not included in this analysis because after clustering, these variables of the clusters would result in decimals smaller than 1, which are meaningless in this case.

```
k.max <- 15
data1 <- sfcluster
wss <- sapply(1:k.max,
        function(k){kmeans(data1, k, nstart=50,iter.max = 15 )$tot.withinss})
wss
plot(1:k.max, wss,
        type="b", pch = 19, frame = FALSE,
        xlab="Number of clusters K",
        ylab="Total within-clusters sum of squares")
```



After filtering the data, an elbow chart is built in order to determine the number of clusters. According to the chart, x-axis represents the number of clusters K and y-axis shows the total within-clusters sum of squares. When x increases from 3 to 4, y decreases approximately 1,500,000, resulting in the total within-clusters sum decreases approximately 1200, which is relatively a large number compared to the whole dataset and cannot be ignored. By continuously increasing x, the slope is becoming flatter and the result would not become more significant. As a result, the clusters would be most appropriate when K is set to be 4.

km <- kmeans(sfcluster, 4)
km$cluster
km$centers
dist(km$centers)



```
> km$centers
  log_price accommodates bathrooms number_of_reviews review_scores_rating  bedrooms     beds
1  5.227225     3.223387  1.285861          6.784423             95.51480 1.3933416 1.745993
2  4.934625     2.817181  1.133260        142.680617             95.55507 1.1167401 1.486784
3  4.817995     2.522222  1.144444        292.555556             95.60000 0.9888889 1.333333
4  5.033451     3.127930  1.195801         58.994141             95.08105 1.2636719 1.696289
> dist(km$centers)
          1         2         3
2 135.89774
3 285.77292 149.87541
4  52.21223  83.68887 233.56333
```

The first cluster is named “luxury party”. The high price of this kind of residences keeps a large group of customers away so that the number of reviews is small. However, this cluster has the highest number of accommodates as well as bedrooms and beds. Due to this fact, this cluster is suitable for people of larger group and would like to a higher price on accommodation.

The second cluster is named “the average” for having all factors between the highest and the lowest scores. This cluster can be considered as the second attractive for the number of reviews and review scores rank the second among the four clusters. People who are not very sensitive to price could find satisfied residences in this cluster.

The third cluster is named “cheap and popular”. Having the lowest price among the four clusters, this cluster attracts the largest number of residents, which can be seen from the relatively large number of reviews compared to the other three clusters. Due to the number of accommodates, bedrooms, and beds, this cluster is suitable for people who travel in group of two or three and who are not willing to pay a large amount of money on accommodation.

The fourth cluster is named “risk”. Having the second highest price, however, this cluster has the lowest review score rating. Although it could offer a relatively large number of rooms and beds and have other benefits, customer’s decisions may be influenced by the relatively low rating. For residences in this cluster, increasing the rating could be the most efficient way to attract more customers.

Step V: Conclusions (10 points)

I. Write a 3-5 paragraph summary that describes your overall process and experience with this assignment. How could these findings be useful? Who could benefit from the data mining that you have performed here? You already summarized your specific steps in some other parts of the write-up, so focus on the big picture here (do not use the conclusion to simply describe everything you did in the other parts).

Business nowadays is going beyond counts, basic reporting and descriptive analysis. To better analysis a business problem or prepare for decision making, data mining is a useful business analytics method to discover patterns and correlations among different variables in large data set to predict outcomes or to build business model. In this project, we interpret a real-world dataset from Airbnb and we chose San Francisco as our researching object. For the data preparation & exploration step, first thing we did in this project is to have an overview of data by exploring and then extract the necessary data that we need for the following steps. In this case, we decided to replace missing values with median because there are 1387 NAs in our dataset which take up 21% of our total data. Accordingly, we conducted five summary statistic functions to learn more about the relationships between different variables and data distribution situation. In addition, we created five different visualized charts for people who have no knowledge in data mining to have an understandable overview for the data quickly. For example, we created violin plot to check whether cleaning fee is a strong influencer on log_price. Data preparation is essential and useful step for business analysts or data scientists to refine their data before they proceed to next steps. More importantly, business analysts can clarify their research direction with these findings.

In the prediction step, we created a multiple linear regression model with the outcome variable log_price. In this step we included 11 independent variables in our model based on our data preparation findings and common sense. In addition, for categorical variables like property_type and room_type, we converted it to dummy variables in order to include them in our model. Moreover, for categorical variables that include too many levels, we replaced levels that only take up a small percent with “others” to simply our process. With our selected variables, the performance of our model came out pretty good with a really low RMSE ratio. With this accurate model, business analysts can create a pricing recommendation system for hosts or any Airbnb management companies to price their properties precisely based on properties’ condition and features. More importantly, by knowing the price range for a type of property, Airbnb hosts can offer a competitive price for listings and increase their occupancy rate.

In the classification step, we included k-nearest neighbors, Naïve Bayes and Classification Tree as three different tools to interpret different problems. For KNN, we build a model to predict whether a new or fictional rental in San Francisco would have a cleaning fee. By using the similar variables from last step and normalizing the dataset, we successfully built a model to test our new rental and the result showed that our rental shouldn't include a cleaning fee. With this model, hosts can create a new dataframe by putting their rentals' features in to see whether they should charge a cleaning fee. Airbnb customers can also it as a reference to see whether paying for cleaning fee for a property is worthy. For Naïve Bayes, we create four similarly-sized bins using the log-price variable and test which bin will our fictional rental fall into. We label our four bins as "Student Budget", "Below Average", "Above Average", "Pricey Digs" and our fictional rental fell into "student budget" section. The accuracy from our result is relatively low and it might be caused by too many predictors in our model. However, Naïve Bayes' finding in general can benefit marketing team in Airbnb management company especially for product positioning. Also, Airbnb can add this as a filter option to simplify customers rental hunting process. Lastly, we built a classification tree that predict the cancellation policy of a new rental with the outcome of "flexible", "moderate", and "strict". With this classification tree, Airbnb hosts can clearly see distribution situation of cancellation policy under different circumstances. For example, when the review_scores_rating is no more than 90, 44% of listings has the cancellation policy of "moderate". This model helps hosts to decide their cancellation policy based on their specific needs.

Finally, in the clustering step, we perform a k-means analysis to compare any selected list of neighborhoods in San Francisco. According the elbow chart result, we divided our data into 4 different clusters and labeled them as "luxury party", "the average", "cheap and popular", "risk". Each cluster represents various types of listing from the data. The most significant different among the clusters' attribute are log_price, number of accommodates, number of bedrooms so that marketers can focus more on these features when segmenting the market. In other words, with this clustering result, Airbnb marketer can segment their products and conduct customize marketing treatments for listings in different clusters in order to attract customers efficiently.

In conclusion, we can consider ourselves as business analysts or consultants for Airbnb and some hospitality management company and provide them valid suggestions based on our data mining knowledge. With our data preparation, we could make sure our models are based on reliable dataset. With our multiple linear regression model, company monitor if their listing is in reasonable price range. With our classification models, they help Airbnb and hosts to know which category their listings fell into and adjust it based on their needs. With clustering model, it helps marketer from Airbnb or management companies to segment their listings into different types and treat it with specific marketing treatments. By apply these models, Airbnb and hosts would have a deeper understanding of their listings and accordingly can find optimal strategy to maximum their profits. To complete this project, we learnt that not only data mining skills are required but also human judgement is involved.