TEAM MEMBERS

Menghan Lu
Yannick Ramcke
Tianyi Tong
Kairu Wang

Boston University

BOSTON UNIVERSITY

This term project was a requirement for the course
MET AD 699 Data Mining in Summer II
lectured by Professor Gregory Page.

Step I: Data Preparation & Exploration

1) Missing Values

Does your data contain any missing values? 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.

Missing values in data is a common phenomenon in real-world problems, However, there are several methods to handle such circumstances in order to reduce bias and to produce powerful models – without losing too much relevant information. Our data contains a lot of missing values. The first method that came up to mind was to try deleting those rows that contain missing values since there have been number of data points in our dataset. Since the data on-hand corresponding to the rows with some missing data did not show heavy outliers in other measures/variables (i.e. columns), we expected the rows with missing data to be within the general range of values. As a result, the amount of lost information was expected to be (almost) non-material. The codes we used to do this are shown below.

R-Code:

```
Project <- read.csv("train.csv")
ProjectLA <- filter(Project, city == "LA")
is.na(ProjectLA)
#Identification of missing values in data set

ProjectLA <- na.omit(ProjectLA)
View(ProjectLA)
```

After reading the file “train.csv” into RStudio, we set a new dataset “ProjectLA” that only contains data that we will analyze by filtering the data that only focuses on records in Los Angeles (“ProjectLA”). However, there had still been a large number of missing values in the new dataset. Since the missing value will impact the result of the following analysis, we used R-function “na.omit()” to remove the records that contains missing values.

2) Summary Statistics

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

R-Code & Screenshots:

mean(Projectla$review_scores_rating)
[1] 94.31278
sd(Projectla$review_scores_rating)
[1] 7.861555
max(Projectla$review_scores_rating)
[1] 100
min(Projectla$review_scores_rating)
[1] 20
median(Projectla$review_scores_rating)
[1] 97

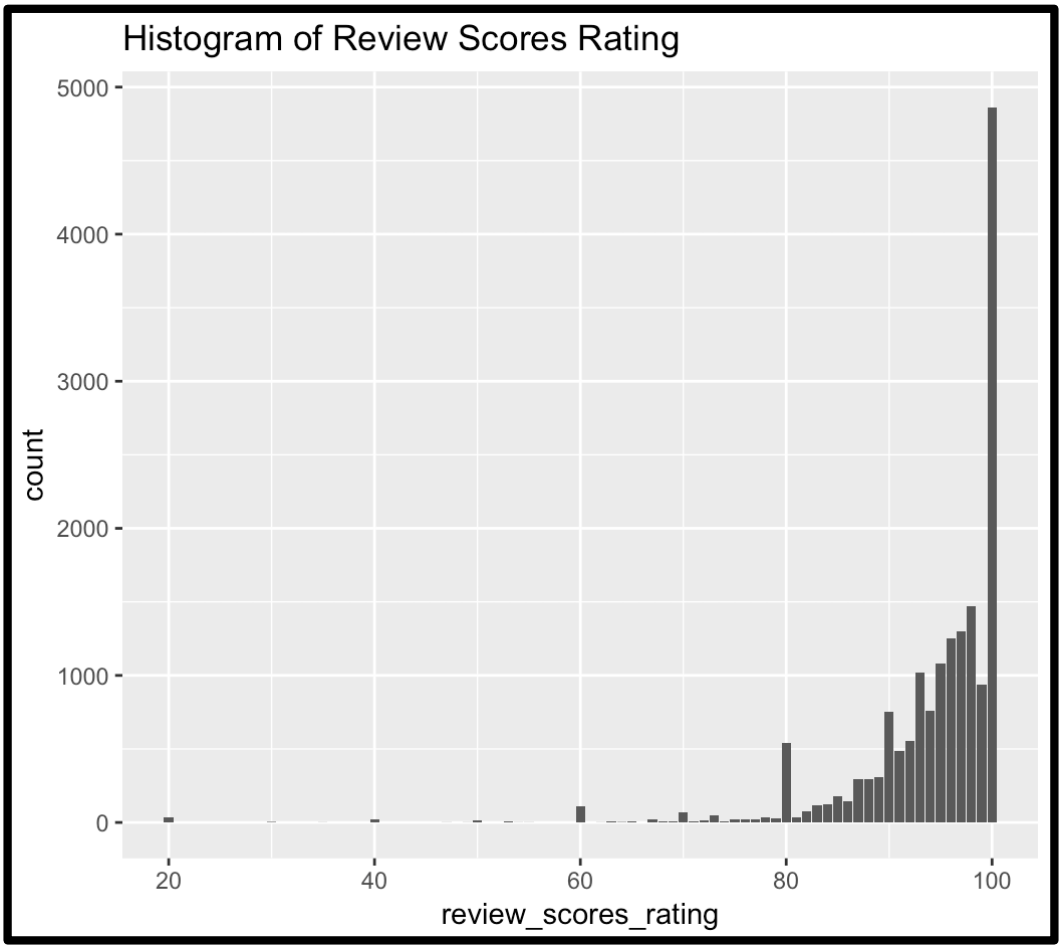
We chose to analyze five summary statistic functions of the review scores rating of the property in LA. The first function we used is **mean()** which returned the result of average rated review scores of LA property that is 94.31. It means the overall situation of all properties of LA are satisfactory. The second function we used is **sd()** which shows the standard deviation of the rated review scores that is 7.86. It is a relatively high standard deviation, which indicates that the data points are spread out over a wider range of values. The data fluctuates substantially over the average rated review scores. We then used the **max()** and **min()** functions to obtain the maximum and minimum of the rated review scores (i.e. the range of review scores). The range between the maximum and minimum is 80, which is significantly large. It means there is a very big disparity of the degree of satisfaction of the properties in LA. However, someone has to keep in mind that these two summary statistics are also very susceptible to outliers. Last but not least, we used **median()** function to find the middle data of the rated review score that is 97, which indicates half of the properties in LA are highly rated. It also hints at the fact that there might be only a few outliers with very low scores.

3) Visualization

Using ggplot, create any 5 plots that help to describe your data (this is intentionally open-ended). Write a two-paragraph description that explains the choices that you made, and what the resulting plots show.

R-Code: Histogram

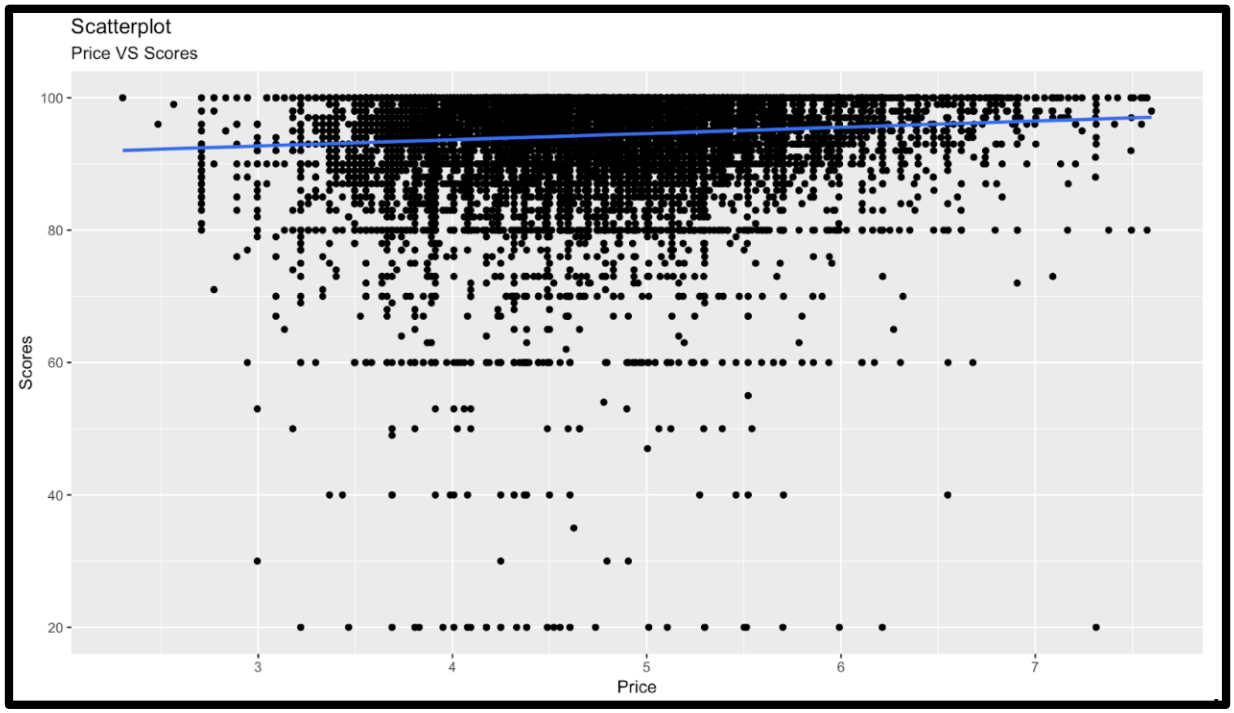
```
ggplot(data = ProjectLA, aes(x = review_scores_rating)) + geom_bar() + ggtitle("Histogram of Review Scores Rating")
```



We then used the **ggplot()** function to plot a histogram of the review scores ratings, which reflect the overall situation of the review scores of all properties in LA. The x-axis represents the review scores and the y-axis represents the frequency of each review scores. It can be seen from the graph that most of the reviews rated the properties very high scores. The most frequent score is 100 and only a very few reviews rated the properties scores that are less than 80, which means the overall situation of properties in LA are in good condition and with high quality. This visualization provides further evidence for the impact of outliers on summary statistics such as minimum values and, to a lesser extent, averages. Since the majority of review scores are relatively high, it can be concluded that the customers' (subjective) expectations are more often than not met by the accommodation in the LA market. It has to be mentioned that these expectations heavily depend on the price paid for using the property. It does not necessarily mean that there are only top-notch properties in the LA market.

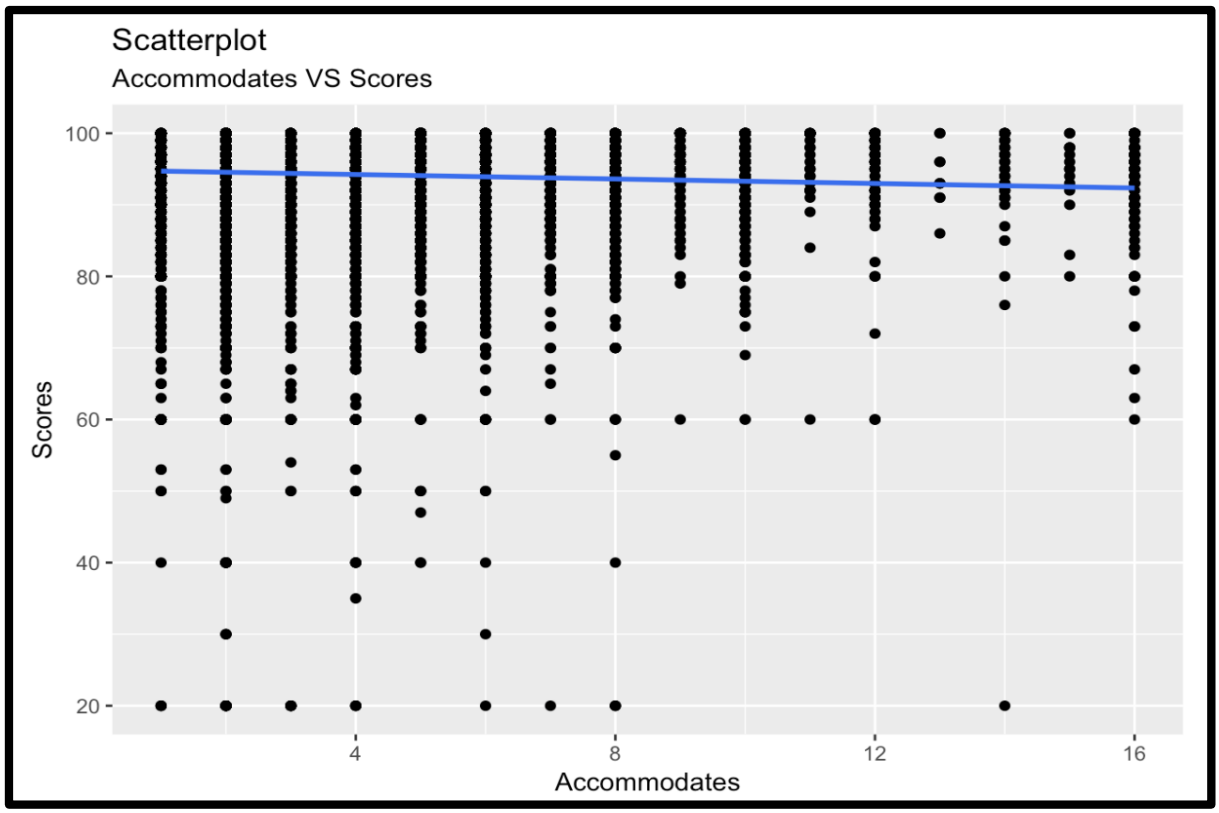
R-Code: Scatterplot I

```
ggplot(ProjectLA, aes(x = log_price, y = review_scores_rating)) + geom_point() +
geom_smooth(method = "lm", se = F) + labs(subtitle = "Price VS Scores", title =
"Scatterplot", x = "Price", y = "Scores")
```



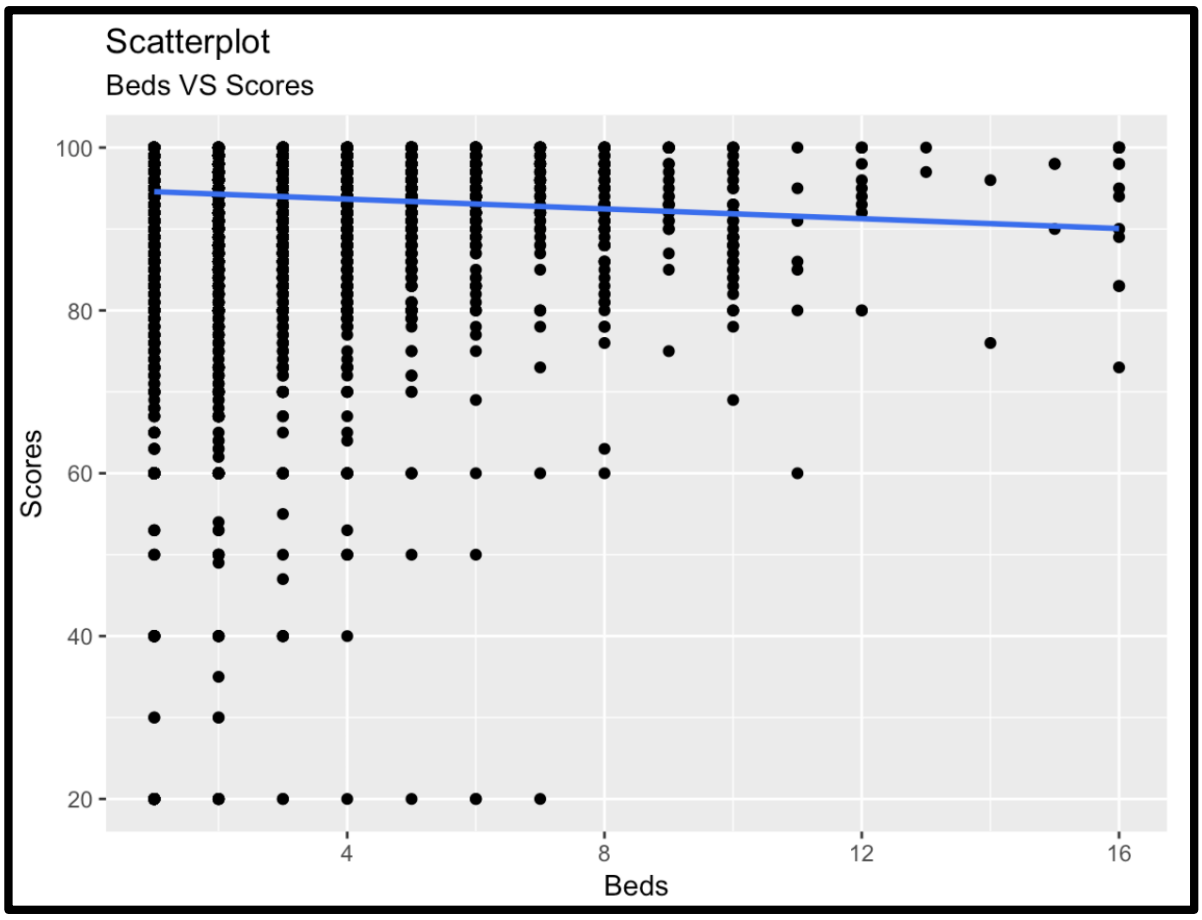
R-Code: Scatterplot II

ggplot(Projectla, aes(x = accommodates, y = review_scores_rating)) + geom_point() + geom_smooth(method = "lm", se = F) + labs(subtitle = "Accommodates VS Scores", title = "Scatterplot", x = "Accommodates", y = "Scores")



R-Code: Scatterplot III

```
ggplot(ProjectLA, aes(x = beds, y = review_scores_rating)) + geom_point() +
geom_smooth(method = "lm", se = F) + labs(subtitle = "Beds VS Scores", title =
"Scatterplot", x = "Beds",  y = "Scores")
```



We used the function **ggplot()** to plot the scatterplots to explore how the variables affect the rated review scores. 'Log price,' 'accommodates' and 'beds' were chosen as the variables that to be analyzed, since they are all numerical and do not need further pre-processing and are easy to interpret.

The relationship between log price and rated review scores is positive. The higher the log price, the higher the rated review scores. The higher price intuitively implies a higher quality of the property, which will be more likely to satisfy the customers and be highly rated – although customers' expectations also change with higher prices and can affect the subjective rating of the property.

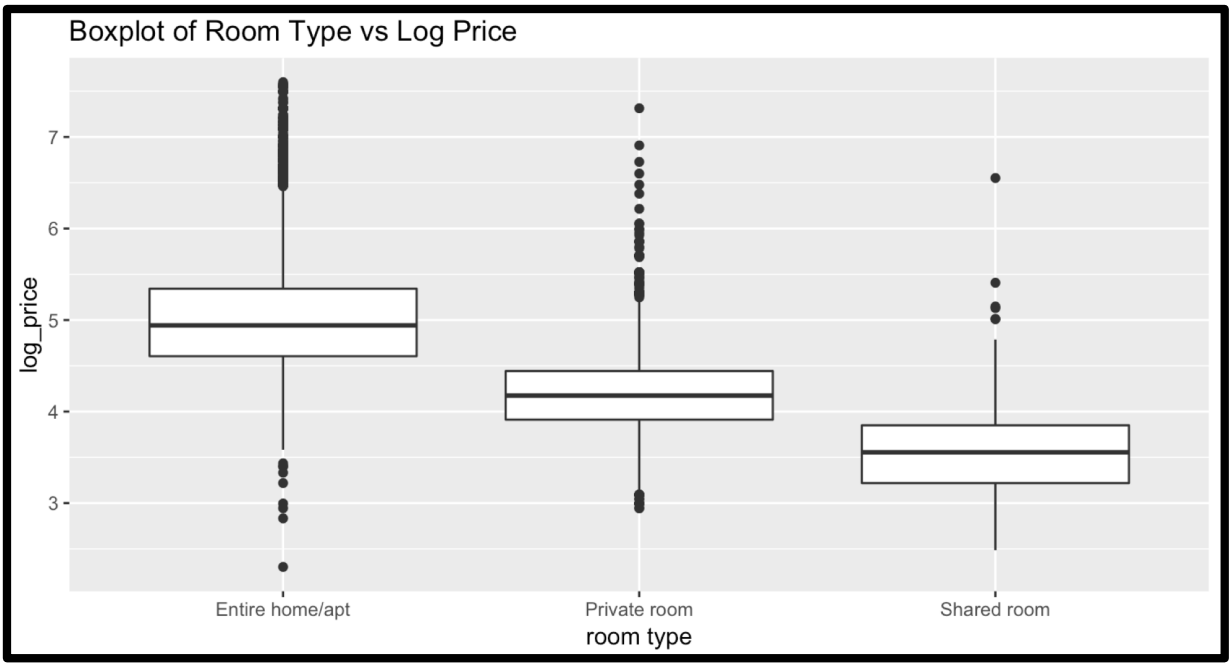
The relationship between accommodates and rated review scores is negative. The properties will get lower review scores if they accommodate more people. Generally, it can be assumed that properties that accommodate many persons are usually used by groups with a lower budget.

The relationship between beds and rated review scores is also negative, which means the properties are not highly rated if they have too many beds. Customers will choose the properties which are quite and private because they don't want to be bothered. A positive relationship in

this instance would have been very surprising given that the number of beds should also be highly correlated with the number of persons that a property can accommodate.

R-Code: Box Plot

```
ggplot(ProjectLA) + geom_boxplot(aes(x = as.factor(room_type), y = log_price)) +
xlab("room type") + ggtitle("Boxplot of Room Type vs Log Price")
```



Last but not least, we used the ***ggplot()*** function to plot a boxplot to analyse the demanded prices segmented by the type of property to get additional insights into how different property types affect how much they cost. The boxplot returns the median, range (incl. 25% & 75% percentiles) as well as outliers outside of said percentiles. The x-axis represents the room/property type and the y-axis represents the log price. The category "entire home/apartment" has the highest median log price but also the largest deviation measured as the range between the 25% and 75% percentiles as well as the complete range measured by the minimum and maximum values. By implication, the prices (i.e. 'log prices') of the group of entire home/apt vary a lot. The 'private room' has the second highest median log price and the least deviation, which means the prices of private rooms are close to each other. The shared room has the lowest median log price and comparatively high deviation. As alluded to before, since the 'property/room type' is not a numerical value, the data had to be pre-processed in order to be plotted in a meaningful manner. However, the ggplot () function offers a built-in parameter to do that quickly by interpreting the values as factors. It can be concluded that properties are priced according what they offer to the customer – what makes total sense. Demanding a price at top of the "Entire Apartment" – price range for a "Shared Room" would probably not generate much demand for that property.

Step II: Data Prediction

1) Multiple Regression with "Review Scores Rating" as Outcome Variable

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.

Before building a regression model, we excluded some variables which are generally hard to affect (or in a meaningful manner) the outcome variable, being the review scores rating. We dropped "id", "city", "description", "first review", "host since", "last review", "latitude", "longitude", "name", "neighbourhood", "thumbnail url", "property type", "amenities", and "zip code". Such columns are some basic information that probably only merely affect the outcome variables, if at all (e.g. ID). On the other hand, other independent variables such as "log price", "accommodates", "bathrooms", "cleaning fee", "host response rate", "number of reviews", "bedrooms", "beds", "host identity verified", and "instant bookable" are much more likely to have a material impact on the review score (i.e. dependent variable). In order to be able to possibly incorporate all aforementioned measures (i.e. columns) into a regression model, the "clean fee", "host response rate", "host identity verified", and "instant bookable" have to be converted into numerical variables (i.e. binary values).

R-Code: Converting Character into Numerical Variables

```
> projectla$cleaning_fee [projectla$cleaning_fee == "True"] <- 1
> projectla$cleaning_fee [projectla$cleaning_fee == "False"] <- 0
> projectla$cleaning_fee <- as.numeric(projectla$cleaning_fee, fixed = TRUE)
> projectla$host_response_rate <- as.numeric(sub("%", "", projectla$host_response_
rate, fixed = TRUE))/100
> projectla$host_identity_verified [projectla$host_identity_verified == "t"] <- 1
> projectla$host_identity_verified [projectla$host_identity_verified == "f"] <- 0
> projectla$host_identity_verified <- as.numeric(projectla$host_identity_verified,
fixed = TRUE)
> projectla$instant_bookable [projectla$instant_bookable == "t"] <- 1
> projectla$instant_bookable [projectla$instant_bookable == "f"] <- 0
> projectla$instant_bookable <- as.numeric(projectla$instant_bookable, fixed = TRU
E)
```

However, we could not be sure that all the possible independent variables have a strong and/or meaningful correlation with the outcome variable.

Therefore, we used a search algorithm (e.g. Forward selection, Backward elimination and Stepwise regression) to find the best model. Specifically, we used the "Backward Stepwise" to get the best fit regression model, selecting the most meaningful variables.

R-Code: Variable Selection [Step I]

```
> fit <- lm(projectla$review_scores_rating ~ log_price + accommodates + bathrooms+
cleaning_fee+host_response_rate+number_of_reviews+bedrooms+beds+host_identity_veri
fied+instant_bookable, data = projectla)
> fit.step <- step(fit, direction = "backward")
Start:  AIC=37832.28
projectla$review_scores_rating ~ log_price + accommodates + bathrooms +
    cleaning_fee + host_response_rate + number_of_reviews + bedrooms +
    beds + host_identity_verified + instant_bookable

                          Df Sum of Sq    RSS   AIC
- number_of_reviews        1       3.0 430462 37830
- cleaning_fee             1      66.4 430526 37832
<none>                                 430459 37832
- host_identity_verified   1     128.3 430588 37833
- bathrooms                1     195.4 430655 37835
- beds                     1     330.7 430790 37838
- bedrooms                 1     566.9 431026 37844
- accommodates             1    2753.1 433213 37894
- instant_bookable         1    3693.6 434153 37916
- host_response_rate       1    5694.5 436154 37963
- log_price                1    7641.7 438101 38007
```

The number of reviews seems to be the least meaningful variable when predicting the overall review score of a property. As a result, this variable is dropped.

R-Code: Variable Selection [Step II & III]

```
Step:  AIC=37830.35
projectla$review_scores_rating ~ log_price + accommodates + bathrooms +
    cleaning_fee + host_response_rate + bedrooms + beds + host_identity_verified +
    instant_bookable

                          Df Sum of Sq    RSS   AIC
- cleaning_fee             1      67.5 430530 37830
<none>                                 430462 37830
- host_identity_verified   1     135.7 430598 37832
- bathrooms                1     197.1 430660 37833
- beds                     1     329.9 430792 37836
- bedrooms                 1     564.9 431027 37842
- accommodates             1    2750.1 433213 37892
- instant_bookable         1    3695.0 434158 37914
- host_response_rate       1    5771.2 436234 37962
- log_price                1    7674.6 438137 38006

Step:  AIC=37829.93
projectla$review_scores_rating ~ log_price + accommodates + bathrooms +
    host_response_rate + bedrooms + beds + host_identity_verified +
    instant_bookable

                          Df Sum of Sq    RSS   AIC
<none>                                 430530 37830
- host_identity_verified   1     121.4 430651 37831
- bathrooms                1     182.6 430713 37832
- beds                     1     306.5 430837 37835
- bedrooms                 1     569.3 431099 37841
- accommodates             1    2865.3 433395 37895
- instant_bookable         1    3695.4 434225 37914
- host_response_rate       1    5713.9 436244 37961
- log_price                1    7633.9 438164 38005
```

In step II, the variables "Cleaning Fee" is dropped give its low impact on the dependent variable. In step III, the variable with the least meaningful impact would have been "Host Identity Verified"; However, the step() - function suggested the 8-variable regression model as the most suited one to predict the dependent variable -- not losing any more (meaningful) information or predictive power.

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

R-Code: Summary Table of the Best Fit Regression - Model

```
> summary(fit.step)

Call:
lm(formula = projectla$review_scores_rating ~ log_price + accommodates +
    bathrooms + host_response_rate + bedrooms + beds + host_identity_verified +
    instant_bookable, data = projectla)

Residuals:
    Min      1Q  Median      3Q     Max
-76.266  -1.974   1.559   3.927  17.448

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            81.97853    0.78381 104.589  < 2e-16 ***
log_price               1.77223    0.13270  13.355  < 2e-16 ***
accommodates           -0.48437    0.05920  -8.182 3.13e-16 ***
bathrooms              -0.26170    0.12670  -2.066 0.038898 *
host_response_rate     669.69075   57.96055  11.554  < 2e-16 ***
bedrooms                0.43184    0.11840   3.647 0.000267 ***
beds                   -0.20569    0.07686  -2.676 0.007459 **
host_identity_verified  0.26514    0.15746   1.684 0.092236 .
instant_bookable       -1.33097    0.14324  -9.292  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.542 on 10059 degrees of freedom
Multiple R-squared:  0.05092,   Adjusted R-squared:  0.05016
F-statistic: 67.46 on 8 and 10059 DF,  p-value: < 2.2e-16
```

This multiple regression model has eight independent variables: Thereby, positive correlations with outcome variables can be observed for "log price", bathrooms", "host response rate", "bedrooms", and "host identity verified. Negative correlations with outcome variables are identified for "accommodates", "beds", and "instant bookable". The model's intercept is 81.98. The equation of the regression model is shown below.

The regression equation is:

$$\boldsymbol{Review\ Score} =$$

$$81.98 + (1.77 * Price) - (0.48 * Accommodates) - (0.26 * Bathrooms) + (669.69 * Host\ Response\ Rate) + (0.43 * Bedrooms) - (0.21 * Beds) + (0.27 * Host\ Identity\ Verified) - (1.33 * Instant\ Booking)$$

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

R-Code & Regression Model - Output:

```
> library(forecast)
> fit.step.pred <- predict(fit.step, projectla)
> accuracy(fit.step.pred, projectla$review_scores_rating)
                     ME     RMSE      MAE        MPE     MAPE
Test set 3.493403e-12 6.539282 4.305397 -0.8231248 5.174229
```

In our regression model, the root mean square error is 6.54.

According to our studies, the RMSE is the square root of the variance of the residuals, which is a major measurement of testing the accuracy of a regression model to predict the response. The RMSE of this multiple regression model is 6.539282, which means the square root of the variance of the residuals of this multiple regression model is 6.539282. RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the dependent variable. Lower RMSE means better regression model we get. Since RMSE is an absolute value, it might be difficult to interpret.

Step III: Classification

1) K-nearest Neighbors

Using k-nearest neighbors, predict whether a rental in your city that accommodates 4 people, has 2 bathrooms, and a strict cancellation policy would have a cleaning fee.

R-Code: Pre-Processing of Data & Fictional Airbnb - Listing

```
library(GGally)
library(dplyr)
library(caret)
library(FNN)
project<-read.csv("train.csv")

#Selecting data we want
projectla<- subset(project, project$city=="LA",
select = c(accommodates,bathrooms,
cancellation_policy,cleaning_fee))

#Omitting missing data
projectla<-na.omit(projectla,invert=FALSE)

#Turning into Dummy Variables
dum<-dummyVars("~cancellation_policy", data=projectla)
projectla<-cbind(projectla, predict(dum,newdata=projectla))

#Converting into Factors
projectla$cleaning_fee<-as.factor(projectla$cleaning_fee)

#Reordering the dataset
projectla<-projectla[,-3]
projectla<-projectla[,c(1,2,4,5,6,7,8,3)]
summary(projectla)
projectla<-projectla[,-6]

#Partitioning Data into Traning & Validation Set
set.seed(111)
samplela<-sample_n(projectla, dim(projectla)[1])
training<-slice(samplela,1:(0.6*dim(samplela)[1]))
validation<-slice(samplela,(0.6*dim(samplela)[1]+1):(dim(samplela)[1]))
```

#Creating Fictional Airbnb - Listing data

myrental<-data.frame(accommodates=4, bathrooms=2,
cancellation_policy.flexible=0,
cancellation_policy.moderate=0, cancellation_policy.strict=1,
cancellation_policy.super_strict_60 =0)

Screenshot:

```
> library(GGally)
> library(dplyr)
> library(caret)
> library(FNN)
> project<-read.csv("train.csv")
> #select data we want
> projectla<- subset(project, project$city=="LA",
+                    select = c(accommodates,bathrooms,
+                               cancellation_policy,cleaning_fee))
> #missing data omit
> projectla<-na.omit(projectla,invert=FALSE)
> #turns into Dummy Variables
> dum<-dummyVars("~cancellation_policy", data=projectla)
> projectla<-cbind(projectla, predict(dum,newdata=projectla))
> #asfactor
> projectla$cleaning_fee<-as.factor(projectla$cleaning_fee)
> #reorder the dataset
> projectla<-projectla[,-3]
> projectla<-projectla[,c(1,2,4,5,6,7,8,3)]
> #summary(projectla)
> projectla<-projectla[,-6]
>
> #partitioning
> set.seed(111)
> samplela<-sample_n(projectla, dim(projectla)[1])
> training<-slice(samplela,1:(0.6*dim(samplela)[1]))
> validation<-slice(samplela,(0.6*dim(samplela)[1]+1):(dim(samplela)[1]))
>
> #selected data
> myrental<-data.frame(accommodates=4, bathrooms=2, cancellation_policy.flexible=0,
+                      cancellation_policy.moderate=0, cancellation_policy.strict=1,
+                      cancellation_policy.super_strict_60 =0)
```

R-Code: Normalization & Partitioning of Data

#Normalization
prolanorm<-projectla
trainnorm<-training
validnorm<-validation
norm<-preProcess(training[,1:6],method=c("center","scale"))
trainnorm[,1:6]<-predict(norm,training[,1:6])
validnorm[,1:6]<-predict(norm,validation[,1:6])
prolanorm[,1:6]<-predict(norm,projectla[,1:6])
myrentnorm<-predict(norm,myrental)

#KNN have to be numeric
nn<-knn(train = trainnorm[,1:6],test = myrentnorm,cl=trainnorm[,7],k=3)
rw<-row.names(training)[attr(nn, "nn.index")]
#[1] "1313" "1533" "2333"

training[rw,]$cleaning_fee
#[1] TRUE TRUE TRUE
#vote for TRUE

Screenshot:

```
> #normalization
> prolanorm<-projectla
> trainnorm<-training
> validnorm<-validation
> norm<-preProcess(training[,1:6],method=c("center","scale"))
> trainnorm[,1:6]<-predict(norm,training[,1:6])
> validnorm[,1:6]<-predict(norm,validation[,1:6])
> prolanorm[,1:6]<-predict(norm,projectla[,1:6])
> myrentnorm<-predict(norm,myrental)
>
> #knn- have to be numeric
> nn<-knn(train = trainnorm[,1:6],test = myrentnorm,cl=trainnorm[,7],k=3)
> rw<-row.names(training)[attr(nn, "nn.index")]
> #[1] "1313" "1533" "2333"
> training[rw,]$cleaning_fee
[1] TRUE TRUE TRUE
Levels: FALSE TRUE
> #[1] TRUE TRUE TRUE
> #vote for TRUE
```

R-Code: Assessment of K-Nearest Neighbor – Model

#Assessing Accuracy for k (=Number of Neighbors)
loop<-round(sqrt(dim(trainnorm)[1]))
kaccuracy<- data.frame(k = seq(1, loop, 1), accuracy = rep(0, loop))

#Computing KNN for different k on Validation Data
for (i in 1:loop)
{ **nnaccuracy** <- knn(trainnorm[,1:6], validnorm[,1:6], cl=trainnorm[,7], k=i)
kaccuracy[i, 2] <- confusionMatrix(nnaccuracy, validnorm[, 7])$overall[1] }
View(kaccuracy)

#Identifying the k – value with the highest Accuracy
row.names(kaccuracy)[kaccuracy$accuracy == max(kaccuracy$accuracy)]

#Re-doing KNN – Model with k=5
nn<-knn(train = trainnorm[,1:6],test = myrentnorm,cl=trainnorm[,7],k=5)
rw2<-row.names(training)[attr(nn, "nn.index")]

training[rw2,]$cleaning_fee
#[1] TRUE TRUE TRUE TRUE FALSE
#vote for TRUE

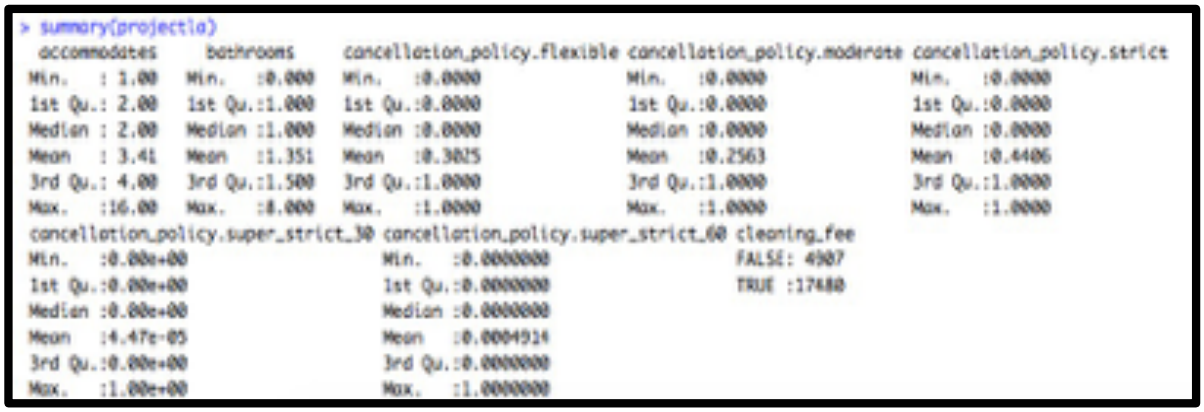
Screenshot:

```
> kaccuracy<- data.frame(k = seq(1, loop, 1), accuracy = rep(0, loop))
> #compute knn for different k on validation.
> #confusionMatrix(actual, predicted, cutoff = 0.5)
> for(i in 1:loop ){
+   nnaccuracy <- knn(trainnorm[,1:6], validnorm[,1:6], cl=trainnorm[,7], k=i)
+   kaccuracy[i, 2] <- confusionMatrix(nnaccuracy, validnorm[, 7])$overall[1]
+ }
> View(kaccuracy)
> #the k value with the highest accuracy
> row.names(kaccuracy)[kaccuracy$accuracy == max(kaccuracy$accuracy)]
[1] "17"
>
> #re-do k=5
> nn<-knn(train = trainnorm[,1:6],test = myrentnorm,cl=trainnorm[,7],k=5)
> rw2<-row.names(training)[attr(nn, "nn.index")]
> training[rw2,]$cleaning_fee
[1] TRUE TRUE TRUE TRUE TRUE
Levels: FALSE TRUE
> #[1] TRUE  TRUE  TRUE  TRUE  FALSE
> #vote for TRUE
```

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

First, we created a new dataframe ("projectla") that only contains required information of the LA market, which are in the columns 'accommodates' , 'bathrooms' ,'cancellation_policy' and 'cleaning_fee.' Regarding the missing data, we can just get rid of them, since there is less than 0.5% missing data of total data available in the entire data set, which will not have any impact on the prediction outcome based on the reasoning mentioned before. In order to make the selected data prepared for KNN, we converted factor data in 'cancellation_policy' to binary dummy variables, and logical data in 'cleaning_fee' to factor data with two levels. By using the **summary()** – function to take a look of our data, we can notice that the observation 'super strict' for the measure 'cancellation policy' happens so rare: 30 observations in total. We decided to drop this column, since it was most likely not relevant for our purposes. Then, we split the selected, pre-processed dataset into two groups -- 60% for training and 40% for validation. We also did normalization for the selected dataset, training data, validation data and the dataset we want to test, so that they are fully prepared for the future use. We took k = 3 for the first time to predict whether we have to pay the cleaning fee under any given situation. It turned out there are three TRUEs in total, so we did vote for TRUE (which means we have to pay the cleaning fee).

Screenshot: Summary Data for "ProjectLA"



```
> summary(projectla)
  accommodates     bathrooms     cancellation_policy.flexible cancellation_policy.moderate cancellation_policy.strict
 Min.   : 1.00   Min.   :0.000   Min.   :0.0000               Min.   :0.0000               Min.   :0.0000
 1st Qu.: 2.00   1st Qu.:1.000   1st Qu.:0.0000               1st Qu.:0.0000               1st Qu.:0.0000
 Median : 2.00   Median :1.000   Median :0.0000               Median :0.0000               Median :0.0000
 Mean   : 3.41   Mean   :1.351   Mean   :0.3025               Mean   :0.2563               Mean   :0.4406
 3rd Qu.: 4.00   3rd Qu.:1.500   3rd Qu.:1.0000               3rd Qu.:1.0000               3rd Qu.:1.0000
 Max.   :16.00   Max.   :8.000   Max.   :1.0000               Max.   :1.0000               Max.   :1.0000
 cancellation_policy.super_strict_30 cancellation_policy.super_strict_60 cleaning_fee
 Min.   :0.00e+00                    Min.   :0.0000000                   FALSE: 4987
 1st Qu.:0.00e+00                    1st Qu.:0.0000000                   TRUE :17488
 Median :0.00e+00                    Median :0.0000000
 Mean   :4.47e-05                    Mean   :0.0004914
 3rd Qu.:0.00e+00                    3rd Qu.:0.0000000
 Max.   :1.00e+00                    Max.   :1.0000000
```

In order to determine which k value we should choose, we tried running the loop from 1 to 116 (equals to the square root of the size of the training set) to get the overall accuracy of the KNN – model, by using the validation data to compare the accuracy of 116 possible k values, as measured against the model built on the training data. As the result shows, the first optimal k value appeared is 110. Since it's more time-consuming and will miss local structure when k = 110 to get the result, we opted for taking k = 5, whose accuracy was almost the same as the optimal one (i.e. k = 110). We re-did the KNN – model to predict whether to pay the cleaning fee with k = 5: The result has been four TRUEs out of five (which means we have to pay the cleaning fee). It turns out that when k = 5, we can get the same prediction result as k = 3 with a higher accuracy, thus k = 5 will be a better choice.

	k	accuracy
1	1	0.7072817
2	2	0.6048693
3	3	0.7161045
4	4	0.6932097
5	5	0.7807684
6	6	0.7803216
7	7	0.7812151
8	8	0.7811034
9	9	0.7811034
10	10	0.7824436
11	11	0.7832254
12	12	0.7722805
13	13	0.7814385

The superior accuracy is shown on the table above: 0.71 (k = 3) vs. 0.78 (k = 5). In order to not lose to much of the local / micro structure, we opted against choosing a higher k – values as even doubling the value of k (i.e. k = 10) would have resulted in only minute improvements in terms of accuracy.

2) Naive Bayes/Classification Trees

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).

R-Code: Exploration & Pre-Processing of Data

#Installation of Required Packages

```
library(dplyr)
library(rpart)
library(rpart.plot)
library(adabag)
library(e1071)
library(caret)
```

#Pre-Processing of Data

```
HousingData <- read.csv(file = "Train.csv")
HousingDataLA <- filter(HousingData, HousingData$city == "LA")
HousingDataLA.Variables <- HousingDataLA[ -c(5, 12, 13, 17, 19, 20, 21, 22, 23, 26, 27)]
```

#Exploratory Analysis of Range of Prices

```
summary(HousingDataLA) #Number of LA - Records: 22453
```

```
summary(HousingDataLA$log_price)
```

```
> summary(HousingDataLA$log_price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.303   4.248   4.605   4.720   5.136   7.600
```

```
max(HousingDataLA$log_price) #7.600402
min(HousingDataLA$log_price) #2.302585
```

```
> max(HousingDataLA$log_price) #7.600402
[1] 7.600402
> min(HousingDataLA$log_price) #2.302585
[1] 2.302585
```

#Categorizing Prices into four similarly-sized Bins with Individual Labels
HousingDataLA.Variables$log_price_bins <- cut(HousingDataLA.Variables$log_price, 4, labels = c("Pricey Digs", "Above Average", "Below Average", "Student Budget"))

#Reordering Columns for Visual Reasons
HousingDataLA.Variables <- HousingDataLA.Variables[,c(1, 2, 19, 3:18)]

#Transforming ID-Column into Row Names
rownames(HousingDataLA.Variables) <- HousingDataLA.Variables$id
HousingDataLA.Variables <- HousingDataLA.Variables[, -c(1)]

	log_price	log_price_bins	property_type	room_type	accommodates	bathrooms	bed_type	cancellation_policy	c
11825529	4.418841	Above Average	Apartment	Entire home/apt	3	1.0	Real Bed	moderate	T
13971273	4.787492	Above Average	Condominium	Entire home/apt	2	1.0	Real Bed	moderate	T
5385260	3.583519	Pricey Digs	House	Private room	2	1.0	Real Bed	moderate	T
17423675	5.010635	Below Average	House	Entire home/apt	4	1.5	Real Bed	strict	T
14066228	4.248495	Above Average	Apartment	Private room	2	1.0	Real Bed	flexible	T

(Excerpt only: columns & rows)

Using any five categorical predictors, build a model using either a naive bayes or classification tree algorithm. Describe a fictional apartment, and use your model to predict which bin it will fall into.

R-Code: Additional Pre-Processing of Data

#Creating Relevant Data Frame with selected Predictor Variables
Selected.Predictors = c("log_price_bins", "property_type", "accommodates", "cancellation_policy", "number_of_reviews", "review_scores_rating")

HousingDataLA.Variables.Relevant = HousingDataLA.Variables[colnames(HousingDataLA.Variables) %in% Selected.Predictors]
#Property Type [3], Accomodates [5], Cancellation Policy [8], Number of Reviews [15] & Review Score Rating [16] selected as predictor variables

#Omitting Records with missing Data (i.e. N/A)
ProjectLA <- na.omit(HousingDataLA.Variables.Relevant)

	log_price_bins	property_type	accommodates	cancellation_policy	number_of_reviews	review_scores_rating
11825529	Above Average	Apartment	3	moderate	15	97
13971273	Above Average	Condominium	2	moderate	9	93
5385260	Pricey Digs	House	2	moderate	2	90
17423675	Below Average	House	4	strict	29	97
583490	Below Average	Apartment	2	strict	2	100
7264511	Above Average	House	2	strict	26	98
3563677	Above Average	Apartment	4	moderate	73	99

(Excerpt only: rows)

#Creating Training (60%) and Validation (40%) Data for "ProjectLA"
set.seed(111)
train.index <- sample(c(1:dim(ProjectLA)[1]), dim(ProjectLA)[1]*0.6)
TrainData.ProjectLA <- ProjectLA[train.index,] #60% Data for Training
ValidData.ProjectLA <- ProjectLA[-train.index,] #40% Data for Training

R-Code: Creating fictional Airbnb - Listing

```
# C1: Membership -> "Unknown" (to be predicted by models)
log_price_bins <- c("Unknown")
# C2: Property Type -> "Apartment"
property_type <- c("Apartment")
# C3: Accomodates -> 5
accommodates <- c(5)
# C4: Cancellation Policy
cancellation_policy <- c("flexible")
# C5 Number of Review -> 15
number_of_reviews <- c(6)
# C6: Review Score Rating
review_scores_rating <- c(99)
Airbnb_NEW <- data.frame(log_price_bins, property_type, accommodates,
cancellation_policy, number_of_reviews, review_scores_rating)
```

	log_price_bins	property_type	accommodates	cancellation_policy	number_of_reviews	review_scores_rating
1	Unknown	Apartment	5	flexible	6	99

R-Code: Classification via Classifications Tree & Naïves Bayes

```
#Assigning AIRBNB_NEW to a Class based on Classification Tree - Model [Default]
Prediction.Class.CT.NEW <- predict(ClassificationTree, newdata = Airbnb_NEW)
Prediction.Class.CT.NEW
#Default Predict() - Function for Classification Trees returns Probability Predictions
```

```
> Prediction.Class.CT.NEW
  Pricey Digs Above Average Below Average Student Budget
1 0.006841505     0.4025086     0.5781072     0.01254276
```

```
#Assigning AIRBNB_NEW to a Class based on Classification Tree - Model ["Class"]
Prediction.Class.CT.NEW.Class <- predict(ClassificationTree, newdata = Airbnb_NEW,
type = "class")
Prediction.Class.CT.NEW.Class
#Type "Class" of Predict() - Function for Classification Trees returns Absolute Predictions
```

```
> Prediction.Class.CT.NEW.Class <- predict(ClassificationTree, newdata = Airbnb_NEW, type = "class")
> Prediction.Class.CT.NEW.Class
            1
Below Average
Levels: Pricey Digs Above Average Below Average Student Budget
```

#Assigning AIRBNB_NEW to a Class based on Naive Bayes - Model
Prediction.Class.NB.NEW <- predict(NaiveBayes, newdata = Airbnb_NEW)
Prediction.Class.NB.NEW

```
> Prediction.Class.NB.NEW
[1] Above Average
Levels: Pricey Digs Above Average Below Average Student Budget
```

Interestingly both models contradict each other when predicting the category of the fictional Airbnb - listing ("Airbnb_NEW"): The classification tree predicts the fictional apartment to br a "Below Average" accommodation, whereas the Naive Bayes to be an "Above Average" accommodation. In other words, they predict an accommodation that is an apartment ("type"), can accommodate five persons ("accommodates"), has a flexible cancellation policy ("cancellation_policy"), six customer reviews ("number_of_reviews"), and an average score of 99 ("review_Scores_rating") to be priced between either below the average {3.6 | 4.9} or above the average {4.9 | 6.3} for the variable "log_price."

That can be glean from the characteristics of cut() - function in R that was used to created the bins: When the number of bins ("breaks") is specified as a single number, the range of the data is divided into pieces of equal length. The range of data is given by the minimum (2.30) and maximum (7.60) of "Log_Prices." Therefore, each bin covers a range of 1.324 and the number of listings per bin inherently varies across these for groups. Since "Above Average" is the third level - bin, the models predict the fictional listing to be priced between 4.95 and 6.27 based on the five predictor variables.

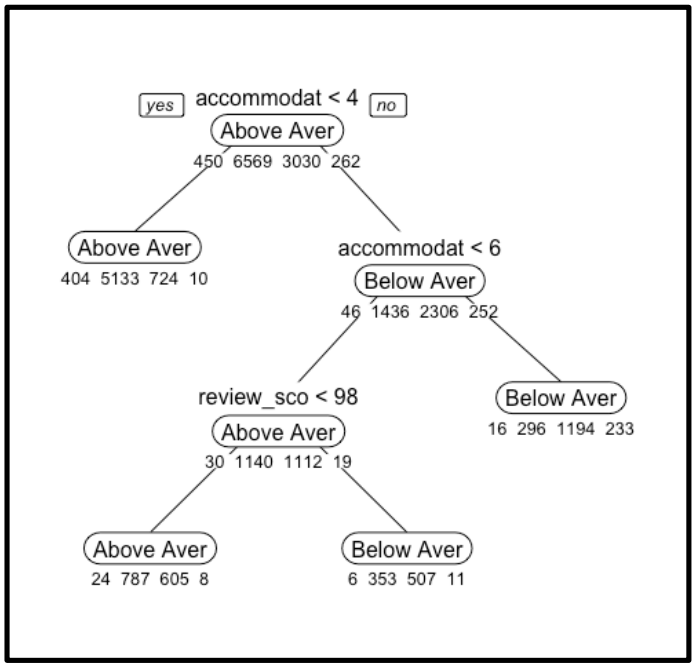
Show a screenshot of the code you used to build your model, the code you used to run the algorithm, and code you used to assess the algorithm.

Creating Classification Tree - Model

R-Code: Classification Tree

#Building Predictive Model: Classification Tree
ClassificationTree <- rpart(log_price_bins ~ ., data = TrainData.ProjectLA, method = "class")

#Plotting Classification Tree
prp(ClassificationTree, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10)



#Identifying Importance of individual Parameters used in Classification Tree
ClassificationTree$variable.importance

```
> ClassificationTree$variable.importance
       accommodates review_scores_rating       property_type    number_of_reviews  cancellation_policy
        1213.717424           25.157176           19.662909            5.233342            1.059842
```

#Assessing Classification Tree - Model with Validation Data
train_control <- trainControl(method = "cv", number=10)
FIT1 = train(log_price_bins ~ ., data = TrainData.ProjectLA, method="rpart", trControl = train_control)

Prediction = predict(FIT1, ValidData.ProjectLA)
confusionMatrix(Prediction, ValidData.ProjectLA$log_price_bins)

```
> confusionMatrix(Prediction, ValidData.ProjectLA$log_price_bins)
Confusion Matrix and Statistics

                Reference
Prediction       Pricey Digs Above Average Below Average Student Budget
  Pricey Digs              0             0             0              0
  Above Average          275          3980           862              7
  Below Average           18           418          1143            172
  Student Budget           0             0             0              0

Overall Statistics

               Accuracy : 0.7452
                 95% CI : (0.7347, 0.7554)
    No Information Rate : 0.6397
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4324
 Mcnemar's Test P-Value : NA
```

As a predictive model, the classification tree has an accuracy of 74.52% when tested with the validation data set. Interestingly, the model skews heavily towards predicting the average classifications (i.e. “Below Average” & “Above Average”) although the entire dataset (743, 441), the training data (450, 262), and validation data (293, 179) by all means have some “Pricey Dogs” and “Student Budget.”

Creating Naïve Bayes - Model

R-Code: Naïves Bayes

#Building Predictive Model: Naïve Bayes
NaiveBayes <- naiveBayes(log_price_bins ~ ., data = TrainData.ProjectLA)
NaiveBayes

#Assessing Naive Bayes - Model with Validation Data
Prediction.Class.NB <- predict(NaiveBayes, newdata = ValidData.ProjectLA)
confusionMatrix(Prediction.Class.NB, ValidData.ProjectLA$log_price_bins)

```
> confusionMatrix(Prediction.Class.NB, ValidData.ProjectLA$log_price_bins)
Confusion Matrix and Statistics

                Reference
Prediction       Pricey Digs Above Average Below Average Student Budget
  Pricey Digs             32           79            21              1
  Above Average          251         4089          1020             14
  Below Average           10          224           874            103
  Student Budget           0            6            90             61

Overall Statistics

               Accuracy : 0.7354
                 95% CI : (0.7248, 0.7458)
    No Information Rate : 0.6397
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4083
 Mcnemar's Test P-Value : < 2.2e-16
```

As a predictive model, the Naïve Bayes - Model has an accuracy of 73.54% when tested with the validation data set. Interestingly, the model skews heavily towards predicting the average classifications (i.e. “Below Average” & “Above Average”) although the entire dataset (743, 441), the training data (450, 262), and validation data (293, 179) by all means have some “Pricey Dogs” and “Student Budget.” In this way, it behaves similar to the Classification Tree - Model, while being slightly less accurate.

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.

R-Code: Interpretation of Predictions

 # Interpretation of Predictions

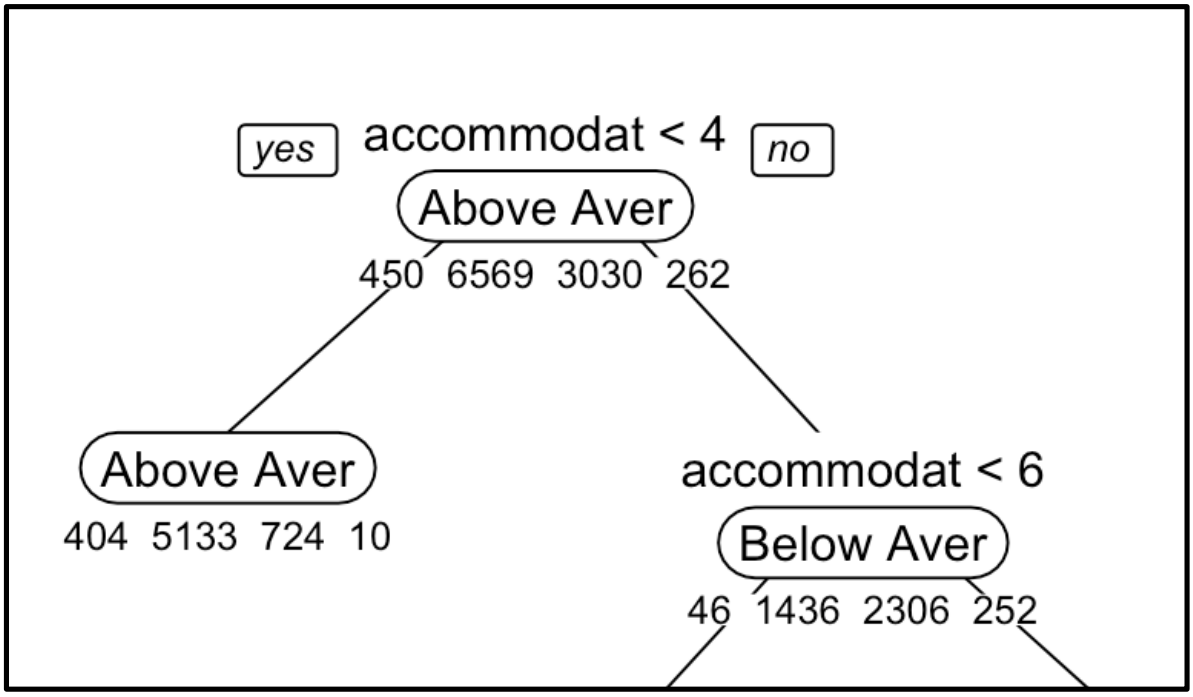
summary(ProjectLA$log_price_bins)
summary(TrainData.ProjectLA$log_price_bins)
summary(ValidData.ProjectLA$log_price_bins)

```
> summary(ProjectLA$log_price_bins)
   Pricey Digs  Above Average  Below Average Student Budget
           743          10967           5035            441
> summary(TrainData.ProjectLA$log_price_bins)
   Pricey Digs  Above Average  Below Average Student Budget
           450           6569           3030            262
> summary(ValidData.ProjectLA$log_price_bins)
   Pricey Digs  Above Average  Below Average Student Budget
           293           4398           2005            179
```

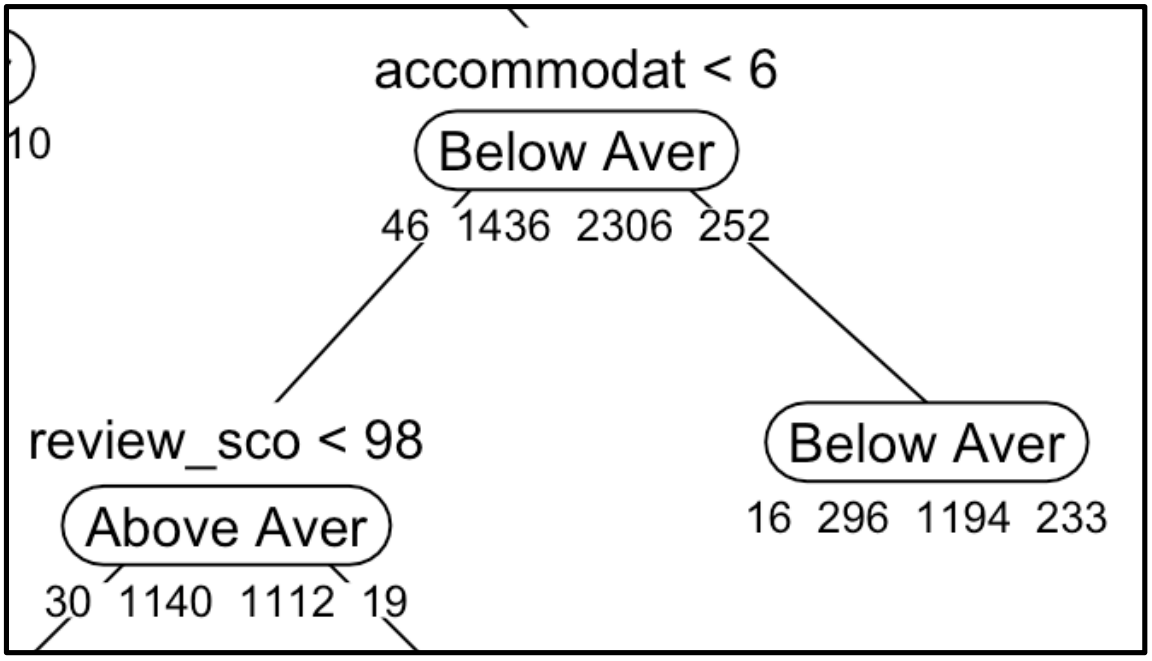
Interestingly, at least the Classification Tree - Model puts paramount importance on the number of accommodates ("accommodates") of the individual listings when classifying / predicting Airbnb - listings.

```
> ClassificationTree$variable.importance
       accommodates review_scores_rating       property_type    number_of_reviews  cancellation_policy
        1213.717424           25.157176           19.662909            5.233342            1.059842
```

Especially "Pricey Dogs" differentiate themselves by accommodating very few persons. However, the Classification Tree - Model uses a cut-off value of 4 accommodates as root node and does not further examines every listing with less than 4 accommodates. By majority decision, the model assigns all listings to the prevalent class among those listing, which is "Above Average" with 5133 listings (versus 404 [Pricey Dogs}, 724 [Below Average], and 10 [Student Budget]).



On the other hand, there are very few "Student Budget" listings comparatively in the entire data set. These listings are characterised by a very high number accommodates. However, even among the listings with more than 6 accommodates "Below Average" - listing dominate that cluster (1194 [Below Average] vs. 233 [Student Budget]) and these listing are predicted to fall under that category.



#The enormous dominance / importance of the "accommodates" - variable calls the variable selection into questions. Therefore, it would be interesting to drop that variable and/or include another variable form the original dataset instead. Indeed, just dropping "accommodates" as predictor variable results in a major increase in the predicted probabilities that a random listing

will be labelled as a “Pricey Dogs” (0.013 -> 0.025) or “Student Budget” (0.006 -> 0.044).

```
> Prediction.Class.CT.NEW
     Pricey Digs Above Average Below Average Student Budget
[1,]  0.04364271     0.6370866     0.2938609     0.02540976
```

Step IV: Conclusions

Write a 3-5 paragraph summary that describes your overall process and experience with this assignment. Think of it as a lab report -- what major types of data mining tasks did you perform, and what did you find? How could these findings be useful? You already summarized your specific steps in some other parts of the write-up, so focus on the big picture here.

In the first step, data preparation & exploration, we omitted the missing values which are contained in our dataset by using "*na.omit()"* function. Then we used five summary statistics functions *mean(), sd(), min(), max() and median()* to evaluate the situation of review scores rating of properties in LA. We can draw a conclusion that the reviews of the LA properties rated high scores with a relatively high standard deviation and large variance. In the visualization process, we used *ggplot()* function to draw scatterplots to explore the how the variables affect the rated review scores. We concluded that the log price and rated review scores has positive relationship, both accommodates and beds have negative relationship with rated review scores. We also used ggplot() function to draw a histogram of the review scores rating and concluded that the overall situation of properties in LA are in good condition and with high quality and Airbnb has a great potential to develop the market in LA. Finally, we used *ggplot()* function to draw a boxplot which indicates different log price should be set according to different room type.

In the second step, we use the backward stepwise to select variables based on the initial multiple regression model, which is assumed all independent variables have strong correlation with the outcome variables, review scores rating. After using the backward stepwise variables selection, the best fit multiple regression model including 8 independent variables, positive correlation with outcome variables are "log price", bathrooms", "host response rate", "bedrooms", "host identity verified, but with negative correlation with outcome variables are "accommodates", "beds", and "instant bookable". Moreover, we use the predict function to check the accuracy of this multiple regression model and get the outcome of RMSE of this multiple regression model is 6.539282. Lower RMSE means better regression model we get. Based on this regression model, it not only can indicate the renters in LA via the Airbnb how to improve their scores review rating efficiently, but also can predict the scores review rating based on the information of those independent variables.

In the third step, regarding K-NN prediction, in pattern recognition, it is a non-parametric method used for classification. By using K-NN model, our selected data is classified according to the majority vote of nearest neighbors selected be the k value. K-NN is one of the most common method used in machine learning. Although it's sensitive to the local structure of the data, it's still an easy and useful way to conduct the prediction.
With regard to the classification by means of a Classification Tree and the Naïve Bayes Method, the enormous impact of the variable selection when predicting new data became obvious. After binning the Airbnb – listing in the LA market into four equally-sized bins based on their price, there inherently have been some listing that were identified as at the lower ("Student Budget")

and upper ("Pricey Digs") limit of the price range. Nonetheless, the classification methods predicted those listings more often than not to belong to the "more average" price category: For example, out of 293 "Pricey Digs" in the validation set, only 32 were indeed correctly predicted to belong to that category by the Classification Tree. However, 251 out of 293 "Pricey Digs" were predicted to be "Above Average" instead.

The lack of accuracy can primarily be attributed to the enormous importance that both models have placed on the number of persons that can be hosted in the property ('accommodates'). However, taking the classification tree as a reference, after the root node, there is no further differentiation for properties that host less than four accommodates and many "Pricey Digs" inevitably default to being predicted as "Above Average" (i.e. majority in data subset of less than four accommodates).

Better incorporating insights from the more exhaustive explanatory analysis in step I & II of the assignment might would have improved the predictor selection and, therefore, the accuracy / predictive power of the models.