Memorandum

From:	**Keon Yeong (Connie) Koh**
To:	**Professor Grant Case**
Date:	**4/13/2021**
Subject:	**Airbnb rental price prediction**

Based on my review of the list of 41,739 airbnb rentals in New York City, I've constructed various models using the dataset supplied to predict the price of a set of Airbnb rentals. Since our goal is to predict price of Airbnb rental price as accurate as possible, root mean squared error (RMSE) metric which measures the average of squared differences between prediction and actual price. I've tried producing lowest RMSE score by leveraging a combination of modeling approaches. The R code has been provided as Appendix 1.

To begin with preparing and exploring the raw Airbnb data, one of the most crucial part with analyzing data was cleaning and make sure there are no NA values in each of the variable. I've used the following functions to begin with cleaning the data:

1) '*glimpse()*' function from dplyr package to get a glimpse of the data, and diagnose which variables are numeric, factor, logical.
2) '*summary(is.na())*' and '*sum.na()*' functions were used to identify how many NA values within each variables of the Airbnb data. There was total 224,969 NA values. Out of 97 variables, there are 18 numeric variables which contain missing values. I've decided to replace these NA values to median (instead of mean).

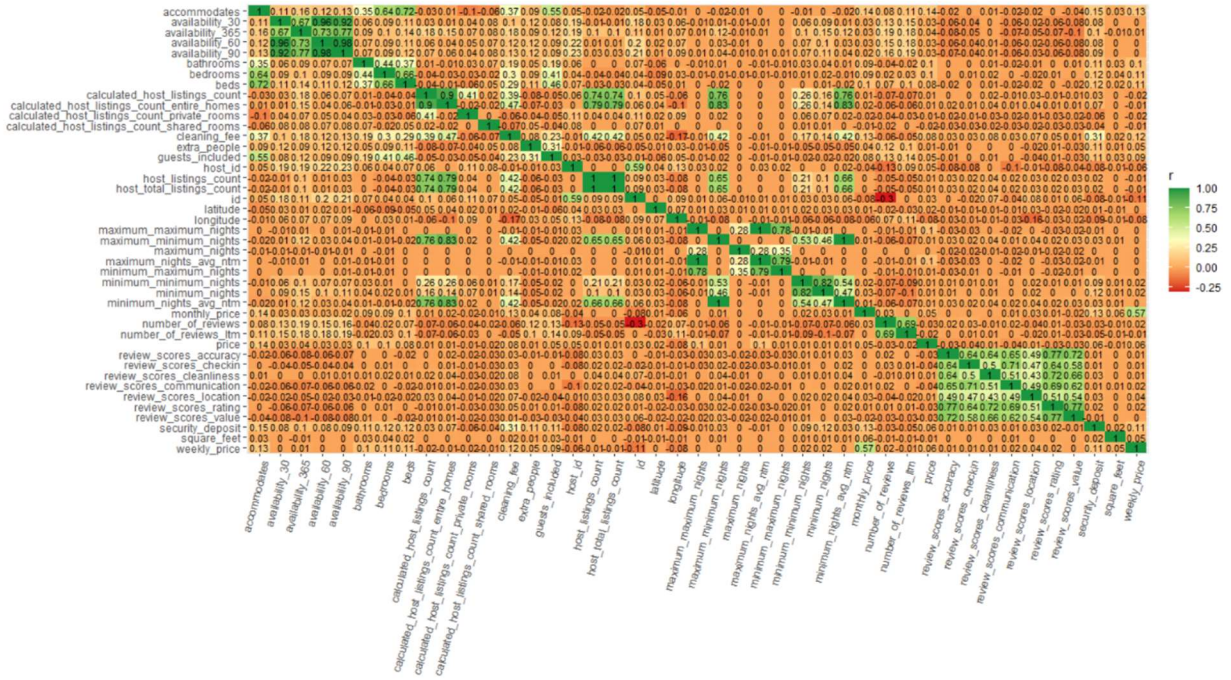
After replacing all missing values to median. I re-used '*sum.na()*' to make sure there are no NA values.

Feature selection:
After the data was cleaned and explored, my first attempt was to use common sense to select variables. I used '*str()*' to glimpse variables and select variables that seem correlated to 'price' variable. Common sense may create bias. I chose to use ranger and tree models as my first model because this model can take both categorical and numeric variables. My first attempt helped reduce the RMSE down to 383.78 with ranger 399.21 with tree which is not much different than the RMSE score from sample.csv data (420.73 RMSE).

Correlation:
After leveraging the first ranger and tree models, I've decided to extract all numeric variables and find which variables are significantly correlated. I created a new variable by using '*!apply()*' function to extract only numeric variables. There are total 44 numeric variables in the data. I then used '*cor(train())*' function to find correlation within numeric variables.

'*Subset*' function was leveraged to see which variables are significantly correlated. However, 44 numeric variables were too much. There was a message in my console, "Selection Algorithm: exhaustive". I used '*corrplot()'* function to visualize the correlation to narrow down the list of correlated variables. I extracted the following numeric variables:

- bedrooms+bathrooms+beds+cleaning_fee+availability_365+extra_people+guests_included+host_listings_count+host_total_listings_count+latitude+maximum_maximum_nights+maximum_minimum_nights+maximum_nights_avg_ntm+minimum_maximum_nights+minimum_minimum_nights+minimum_nights+minimum_nights_avg_ntm+monthly_price+review_scores_cleanliness+review_scores_location+security_deposit+weekly_price.



Lasso proved to be the most efficient at identifying the most important variables. It also eliminated uncorrelated variables. Out of 44 numeric variables, 27 important variables were identified.

Modeling analysis:
Various modeling approaches were leveraged, including tree, ranger, random forest. I started with an RMSE of 420.75 (from sample.csv data), with the goal of reducing it as much as possible as our measure of optimal model.

After iterating variables on various models, the best modeling technique I've found was tuneRanger. A ranger with the 27 important variables reduced the RMSE score way down to 311.47 (whereas tree model reduced to 338.99). It was interesting to see that adding couple of factor variables (room_type, neighbourhood_group_cleansed, host_neighbourhood, property_type, and cancellation_policy) in a ranger model reduced the RMSE score to 291.65. tuneRanger even reduced the RMSE score to 288.64, which was my final RMSE score for this Kaggle project.

What I did right was that cleaning up the date by replacing NA values in the beginning step. I then replaced NA values to both mean and median and iterated models and got better RMSE score with median value. Also, it was really helpful to find the correlated numeric variables by using '*corr()*' function and as well as using lasso feature selection to remove dummy variables. Out of 44 numeric variables, lasso helped to find 27 important variables. What I didn't do right was that I

should have done feature selection in the initial process after cleaning up the data. It was challenging to move back and forth between the data and the modeling. I spent lots of time on just iterating various models without using lasso or feature selection. I was immediately able to reduce RMSE value to down to 311.46 (public RMSE score) with the selected important variables by using lasso.

Going through the process of data cleanup, feature selection and iterating on models was the correct approach. If I was funded for a Phase 2, I would have worked on figuring out more effective ways to cleaning up the data to improve data quality (instead of just replacing NA values to mean or median). This might be helpful for finding out more accurate rental price.

Appendix 1

#Introductory Steps

```
#Install Packages
#Initiate Libraries
install.packages('tidyverse')
install.packages('leaps')
install.packages('caret')
install.packages('ggthemes')
install.packages('glmnet')
install.packages('dplyr')
install.packages('dataiku')
install.packages('ggplot')
library(tidyverse)
library(dplyr)
library(leaps)
library(caret)
library(ggthemes)
library(glmnet)
library(tidyr)
library(ggplot2)

# read dataset
analysisData = read.csv('C://Users/Connie/Documents/APAN5200/kaggle/analysisData.csv', stringsAsFactors = F)
str(analysisData)
summary(analysisData)

# Construct a simple model
model = lm(price~minimum_nights+number_of_reviews,analysisData)

# read in scoring data and apply model to generate predictions
scoringData = read.csv('C://Users/Connie/Documents/APAN5200/kaggle/scoringData.csv')
summary(scoringData)
str(scoringData)
colnames(scoringData)
str(analysisData)
pred = predict(model, newdata=scoringData)

# Construct submission from predictions
submissionFile = data.frame(id = scoringData$id, price = pred)
write.csv(submissionFile, 'C://Users/Connie/Documents/APAN5200/kaggle/sample_submission.csv', row.names=F)

# Add the price column to scoringData because it is not included. Note = we are setting the value to 0
# Create a column name called price. Add the column above to the scoringData with a zero value .Reorder the scoringData dataset to be
id, price, then all everything else that isn't id and price
namevector <- c("price")
scoringData[,namevector] <- 0
scoringData <- scoringData %>%
  select(id,price, everything())
summary(scoringData)

colnames(analysisData)
str(analysisData)
analysisData$zipcode <- as.character(analysisData$zipcode)
scoringData$zipcode <- as.character(scoringData$zipcode)
analysisData$host_is_superhost <- as.logical(analysisData$host_is_superhost)
analysisData$host_has_profile_pic <- as.logical(analysisData$host_has_profile_pic)
analysisData$host_identity_verified <- as.logical(analysisData$host_identity_verified)
analysisData$instant_bookable <- as.logical(analysisData$instant_bookable)
analysisData$require_guest_profile_picture <- as.logical(analysisData$require_guest_profile_picture)
analysisData$require_guest_phone_verification <- as.logical(analysisData$require_guest_phone_verification)
analysisData$is_location_exact <- as.logical(analysisData$is_location_exact)
```

```
## If everything is OK we should only see a "Different number of rows" difference between the two datasets.
all_equal(analysisData, scoringData, convert = TRUE)

#Step 1 Split data
set.seed(5656)
ksplit <- createDataPartition(y = analysisData$price, p=.7, list=F, groups=50)
train <- analysisData[ksplit,]
test <- analysisData[-ksplit,]
nrow(train)
nrow(test)
nrow(analysisData)

#Step2
train$train_test_score <- "train"
test$train_test_score <- "test"
scoringData$train_test_score <- "score"
baseData <- bind_rows(train, test, scoringData)

#Step3 Start of the Data Wrangling Processes
# use glimpse function from dplyr package to get a glimpse of the data
glimpse(baseData)

Summary(is.na(baseData))
baseData$bed_type <- factor(baseData$bed_type)
baseData$property_type <- factor(baseData$property_type)
baseData$instant_bookable <- factor(baseData$instant_bookable)

# Clean data
baseData$bedrooms <- as.numeric(baseData$bedrooms)
baseData$beds <- as.numeric(baseData$beds)
baseData$access <- as.character(baseData$access)
baseData$host_location <- as.factor(baseData$host_location)
baseData$host_neighbourhood <- as.factor(baseData$host_neighbourhood)
baseData$host_verifications <- as.factor(baseData$host_verifications)
baseData$street <- as.factor(baseData$street)
baseData$neighbourhood <- as.factor(baseData$neighbourhood)
baseData$neighbourhood_cleansed <- as.factor(baseData$neighbourhood_cleansed)
baseData$neighbourhood_group_cleansed <- as.factor(baseData$neighbourhood_group_cleansed)
baseData$city <- as.factor(baseData$city)
baseData$room_type <- as.factor(baseData$room_type)
baseData$smart_location <- as.factor(baseData$smart_location)
baseData$cancellation_policy <- as.factor(baseData$cancellation_policy)
baseData$latitude <- as.numeric(baseData$latitude)
baseData$longitude <- as.numeric(baseData$longitude)
baseData$calendar_updated <-as.factor(baseData$calendar_updated)
baseData$amenities <- as.factor(baseData$amenities)
baseData$calendar_updated  <- as.factor(baseData$calendar_updated)

## sum.na() to identify how many na values within baseData. There were total 224,969 NA values within baseData
## Out of 97 variables, there are 16 variables which contain NA values
summary(is.na(baseData))
#Replace NA's in weekly_price variables with median of the remaining values
baseData$bathrooms[is.na(baseData$bathrooms)]<-median(baseData$bathrooms, na.rm=TRUE)
baseData$bedrooms[is.na(baseData$bedrooms)]<-median(baseData$bedrooms, na.rm=TRUE)
baseData$beds[is.na(baseData$beds)]<-median(baseData$beds, na.rm=TRUE)
baseData$weekly_price[is.na(baseData$weekly_price)]<-median(baseData$weekly_price,na.rm=TRUE)
baseData$monthly_price[is.na(baseData$monthly_price)]<-median(baseData$monthly_price,na.rm=TRUE)
baseData$security_deposit[is.na(baseData$security_deposit)]<-median(baseData$security_deposit, na.rm=TRUE)
baseData$cleaning_fee[is.na(baseData$cleaning_fee)]<-median(baseData$cleaning_fee, na.rm=TRUE)
baseData$review_scores_rating[is.na(baseData$review_scores_rating)]<-median(baseData$review_scores_rating, na.rm=TRUE)
baseData$review_scores_accuracy[is.na(baseData$review_scores_accuracy)]<-median(baseData$review_scores_accuracy, na.rm=TRUE)
baseData$review_scores_cleanliness[is.na(baseData$review_scores_cleanliness)]<-median(baseData$review_scores_cleanliness, na.rm=TRUE)
baseData$review_scores_checkin[is.na(baseData$review_scores_checkin)]<-median(baseData$review_scores_checkin, na.rm=TRUE)
baseData$review_scores_communication[is.na(baseData$review_scores_communication)]<-median(baseData$review_scores_communication,
na.rm=TRUE)
```

```
baseData$review_scores_location[is.na(baseData$review_scores_location)]<-median(baseData$review_scores_location, na.rm=TRUE)
baseData$review_scores_value[is.na(baseData$review_scores_value)]<-median(baseData$review_scores_value, na.rm=TRUE)
baseData$reviews_per_month[is.na(baseData$reviews_per_month)]<-median(baseData$reviews_per_month, na.rm=TRUE)
baseData$square_feet[is.na(baseData$square_feet)]<-median(baseData$square_feet, na.rm=TRUE)
baseData$host_listings_count[is.na(baseData$host_listings_count)]<-median(baseData$host_listings_count, na.rm=TRUE)
baseData$host_total_listings_count[is.na(baseData$host_total_listings_count)]<-median(baseData$host_total_listings_count, na.rm=TRUE)
```

#Step 4 - Resplit the data across Train - Test – Score. Must resplit the data (train, test, score) to update RMSE score

```
train <- baseData  %>%
  filter(train_test_score == "train")
test <- baseData  %>%
  filter(train_test_score == "test")
score <- baseData  %>%
  filter(train_test_score == "score")
nrow(analysisData); nrow(train); nrow(test); nrow(score)
```

#Step 5:

```
library(ranger)
library(randomForest)
library(xgboost)
library(tidyverse)
library(rpart)
library(rpart.plot)
library(caret)
library(gbm)
library(vtreat)
library(ROCR)
library(caTools)
```

#Correlation with numeric variables

```
baseData_numeric1 = baseData[,!sapply(baseData, is.character)]
baseData_numeric2 = baseData_numeric1[,!sapply(baseData_numeric1, is.logical)] # 60 factor/integer/numeric only
baseData_numeric3 = baseData_numeric2[,!sapply(baseData_numeric2, is.factor)] # 44 integer/numeric only

glimpse(baseData_numeric3)
set.seed(1031)
split2 = createDataPartition(y=baseData_numeric3$price,p = 0.7,list = F,groups = 100)
train2 = baseData_numeric3[split2,]
test2 = baseData_numeric3[-split2,]
nrow(train2)
nrow(test2)
nrow(baseData_numeric3)

library(ggthemes)
library(corrplot)
cor(train2[,-44])
round(cor(train2[,-44]),4)*100
```

#FORWARD SELECTION # Step: AIC=344704.5

```
start_mod = lm(price~1,data=train2)
empty_mod = lm(price~1,data=train2)
full_mod = lm(price~.,data=train2)
forwardStepwise = step(start_mod,
                       scope=list(upper=full_mod,lower=empty_mod),
                       direction='forward')
summary(forwardStepwise)
```

#BACKWARD SELECTION 344704.5

```
start_mod = lm(price~.,data=train2)
empty_mod = lm(price~1,data=train2)
full_mod = lm(price~.,data=train2)
backwardStepwise = step(start_mod,scope=list(upper=full_mod,lower=empty_mod),direction='backward')
```

#HYBRID SELECTION 344708.5

```
start_mod = lm(price~1,data=train2)
empty_mod = lm(price~1,data=train2)
full_mod = lm(price~.,data=train2)
hybridStepwise = step(start_mod,scope=list(upper=full_mod,lower=empty_mod),direction='both')

summary(hybridStepwise)
summary(forwardStepwise)
summary(backwardStepwise)

hybridStepwisecoeff <- as.data.frame(summary(hybridStepwise)$coefficients)
backwardStepwisecoeff <- as.data.frame(summary(backwardStepwise)$coefficients)
forwardStepwisecoeff <- as.data.frame(summary(forwardStepwise)$coefficients)
```

#LASSO - Investigated the 27 variables out of 44 variables seems correlated.

```
library(broom)
x = model.matrix(price~.-1,data=train2)
y = train2$price
lassoModel = glmnet(x,y,alpha=1)
set.seed(617)
cv.lasso = cv.glmnet(x,y,alpha=0) #10fold cross validation is default
plot(cv.lasso)
coef(cv.lasso)
```

#Final Model 1 with ranger model
#PUBLIC RMSE SCORE 291.65571 (RMSE score 248.5316 (After the competition was closed))

```
forest_ranger3 = ranger(price~room_type+neighbourhood_group_cleansed+host_neighbourhood+property_type+
cancellation_policy+latitude+host_listings_count+host_total_listings_count+accommodates+bathrooms+bedrooms+beds+weekly_price+monthly
_price+security_deposit+cleaning_fee+extra_people+guests_included+maximum_nights+minimum_nights+review_scores_location+maximum_
minimum_nights+minimum_minimum_nights+minimum_maximum_nights+minimum_nights_avg_ntm+maximum_nights_avg_ntm+availabilit
y_365+availability_30+availability_60+availability_90+calculated_host_listings_count+calculated_host_listings_count_entire_homes,
data=train, num.trees = 1000)
pred3 = predict(forest_ranger3, data = test,num.trees = 1000)
rmse_forest_ranger3 = sqrt(mean((pred3$predictions-test$price)^2)); rmse_forest_ranger3
predRangerTest3 <- predict(forest_ranger3, data=score)
submissionFile = data.frame(id = score$id, price = predRangerTest3$predictions)
write.csv(submissionFile, 'C://Users/Connie/Documents/APAN5200/kaggle/03submission.csv', row.names=F)
```

#FINAL MODEL2 - TUNE WITH RANGER
#PUBLIC RMSE SCORE 288.63676 (RMSE score 238.8789() (After the competition was closed)

```
trControl=trainControl(method="cv",number=5)
tuneGrid = expand.grid(mtry=1:4,
                       splitrule = c('variance','extratrees','maxstat'),
                       min.node.size = c(2,5,10,15,20,25))
set.seed(617)
cvModel = train(price~room_type+neighbourhood_group_cleansed+host_neighbourhood+property_type+
cancellation_policy+latitude+host_listings_count+host_total_listings_count+accommodates+bathrooms+bedrooms+beds+weekly_price+monthly
_price+security_deposit+cleaning_fee+extra_people+guests_included+maximum_nights+minimum_nights+review_scores_location+maximum_
minimum_nights+minimum_minimum_nights+minimum_maximum_nights+minimum_nights_avg_ntm+maximum_nights_avg_ntm+availabilit
y_365+availability_30+availability_60+availability_90+calculated_host_listings_count+calculated_host_listings_count_entire_homes, data=train,
method="ranger", num.trees=1000, trControl=trControl,
tuneGrid=tuneGrid)

cv_forest_ranger = ranger(price~room_type+neighbourhood_group_cleansed+host_neighbourhood+property_type+
cancellation_policy+latitude+host_listings_count+host_total_listings_count+accommodates+bathrooms+bedrooms+beds+weekly_price+monthly
_price+security_deposit+cleaning_fee+extra_people+guests_included+maximum_nights+minimum_nights+review_scores_location+maximum_
minimum_nights+minimum_minimum_nights+minimum_maximum_nights+minimum_nights_avg_ntm+maximum_nights_avg_ntm+availabilit
y_365+availability_30+availability_60+availability_90+calculated_host_listings_count+calculated_host_listings_count_entire_homes, data=train,
num.trees = 1000, mtry=cvModel$bestTune$mtry,  min.node.size = cvModel$bestTune$min.node.size, splitrule = cvModel$bestTune$splitrule)
pred = predict(cv_forest_ranger, data =test, num.trees = 1000)
rmse_cv_forest_ranger = sqrt(mean((pred$predictions-test$price)^2)); rmse_cv_forest_ranger
pred_Rangertune <- predict(cv_forest_ranger, data=score)
submissionFile = data.frame(id = score$id, price = pred_Rangertune$predictions)
write.csv(submissionFile, 'C://Users/Connie/Documents/APAN5200/kaggle/10submission.csv', row.names=F)
```