APPLIED DATA ANALYTICS – AIRBNB LISTING PREDICTION

**Analysis on Airbnb data**

This analysis aims to build a predictive model to estimate the successful listing of a property in Airbnb.

**Cleaning the data**

This includes loading the data, understanding its structure and steps used in cleaning the data.

# loading the library  
options(repos = c(CRAN = "https://cran.r-project.org"))  
install.packages("tidyverse")

library("tidyverse")

# reading the data and understanding its structure  
DF=read.csv("C:\\Users\\97152\\OneDrive\\Desktop\\listing\\listings1.csv")  
#str(DF)  
#summary(DF)

# removing unwanted columns to reduce complexity and computational power  
DF = subset(DF, select = -c(listing\_url, scrape\_id,source,name,description ,neighborhood\_overview,picture\_url,host\_url,host\_name,host\_location,host\_about,host\_thumbnail\_url,host\_picture\_url,host\_neighbourhood,host\_listings\_count,host\_total\_listings\_count,host\_verifications,host\_has\_profile\_pic,host\_response\_time,neighbourhood,neighbourhood\_group\_cleansed,bathrooms,bathrooms\_text,bedrooms,beds,latitude,longitude,calendar\_updated,license,calendar\_last\_scraped,number\_of\_reviews,number\_of\_reviews\_l30d,first\_review,last\_review))  
DF = DF[DF$host\_is\_superhost!= "", , drop = FALSE]

# removing null values  
DF = drop\_na(DF,host\_response\_rate,minimum\_minimum\_nights,maximum\_minimum\_nights,minimum\_maximum\_nights,maximum\_maximum\_nights,minimum\_nights\_avg\_ntm,maximum\_nights\_avg\_ntm,review\_scores\_rating,review\_scores\_accuracy,review\_scores\_cleanliness,review\_scores\_checkin,review\_scores\_communication,review\_scores\_location,review\_scores\_value)  
any(duplicated(DF))

## [1] FALSE

#str(DF)

**Data visualization**

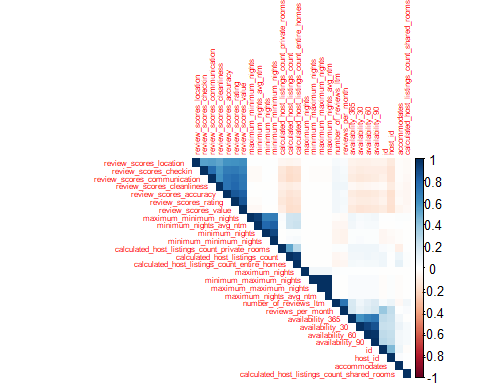
Correlation Matrix

install.packages("corrplot")

library("corrplot")

## corrplot 0.92 loaded

numeric\_columns <- DF[sapply(DF, is.numeric)]  
cor\_matrix <- cor(numeric\_columns)  
par(mar = c(2, 2, 2, 2))   
corrplot(cor\_matrix, method = "color", type = "upper", order = "hclust", tl.cex = 0.5)



The above visualization shows the correction matrix for multiple features of Airbnb dataset. We can see that there are high correlation between review scores, availability, minimum nights and maximum nights. These features should not be used together to avoid multicollinearity.

**Data preprocessing**

-Converting datatypes of features to Date, factor, removing $ and % symbol from numbers and converting them to numeric datatype.

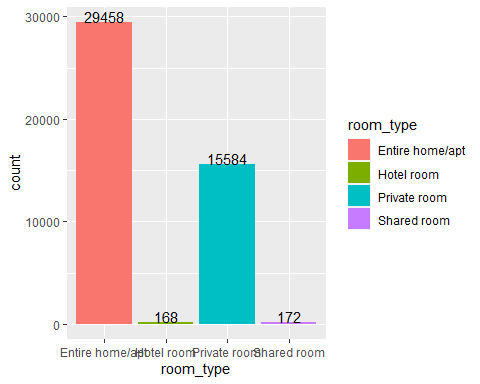
-Dropping the NA values from host\_acceptance\_rate and host\_response\_rate after conversion.

DF$last\_scraped <- as.Date(DF$last\_scraped)  
DF$host\_since <- as.Date(DF$host\_since)  
DF$price <- as.numeric(gsub("[^0-9.]", "", DF$price))  
DF$host\_response\_rate <- as.numeric(gsub("[^0-9.]", "", DF$host\_response\_rate))  
DF$host\_acceptance\_rate <- as.numeric(gsub("[^0-9.]", "", DF$host\_acceptance\_rate))  
DF$host\_is\_superhost <- as.factor(DF$host\_is\_superhost)  
DF$host\_identity\_verified=as.factor(DF$host\_identity\_verified)  
DF$has\_availability=as.factor(DF$has\_availability)  
DF$instant\_bookable=as.factor(DF$instant\_bookable)  
DF = drop\_na(DF,host\_acceptance\_rate,host\_response\_rate)

**Bar chart visualization for Room\_type and count of listings**

install.packages("ggplot2")

library(ggplot2)  
ggplot(DF, aes(x = room\_type, fill = room\_type)) +  
 geom\_bar(stat = "count") +   
 geom\_text(aes(label = after\_stat(count)), stat = "count", vjust = 0, colour = "Black")



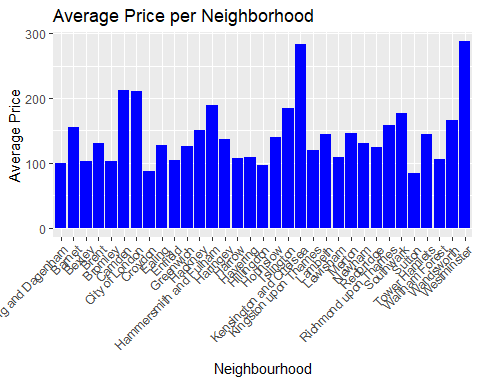
labs(title = "Number of Listings by Room Type",  
 x = "Room Type",  
 y = "Number of Listings")

## $x  
## [1] "Room Type"  
##   
## $y  
## [1] "Number of Listings"  
##   
## $title  
## [1] "Number of Listings by Room Type"  
##   
## attr(,"class")  
## [1] "labels"

The above visualization shows the number of listings for each room type. we can see that there are more number of listings for Entire home/apt and private room but less number of listings for Hotel room and Shared room. This indicated that guests often prefer larger private space than shared space.

**Bar chart visualization for average price across neighborhood**

ggplot(DF, aes(x = neighbourhood\_cleansed, y = price)) +  
 geom\_bar(stat = "summary", fun = "mean", fill = "blue") +  
 labs(title = "Average Price per Neighborhood",  
 x = "Neighbourhood",  
 y = "Average Price")+  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))



The above visualization shows the average price across neighbourhood. There are high prices for westminster, kensington, london and camden. Property prices are higher for these locations as they are close to tourist attractions and have high convenience in reaching the destination.

**Feature Engineering**

1. Calculated host\_experience in days for the listings from data last scraped and since the year host made the listing.
2. Calculated overall\_review\_score as the average of reviews scores on rating, accuracy, cleanliness, check-in, communication, location, and value for money.
3. Calculated the total count of amenities from the list of amenties provided for listings.
4. The limited availability across listings indicates a high level of demand. Based on this availability\_365\_score is calculated which return score-4 when availability for 365 days is less than 100. It means that there is high occupancy.

**Calculating the target variable ‘Good/Bad’**

-Below are features and conditions for calculating the target variable. If the condition is satisfied it assigns a weight 2 otherwise value 1. The sum of these weights is stored in a variable DF$value. Note that for availability\_365\_score, it returns availability\_365\_score itself when positive, 0 otherwise. When the variable DF$value is above a certain threshold it returns the target variable - Good/Bad.

Listings are considered as good listings when they satisfy the below conditions, and they are assigned higher score.

-**Host Experience:** when listings from hosts have more than 3 years of experience in operating and renting accommodations.

-**Host Acceptance Rate:** when hosts have a high acceptance rate above 90%.

-**Amenities Count:** when listings have more than 20 amenities.

-**Room Type:** when listing is an Entire homes/apartments or private room.

-**Availability 365 score:** when availability for 365 has fewer entries.

-**Number of Reviews\_ltm:** when number of reviews for the last 12months is more than 10.

-**Superhost Status:** when the host is a superhost as they are reliable and provide a positive guest experience.

#calculating host experience in airbnb  
DF$host\_experience = as.numeric(difftime(DF$last\_scraped, DF$host\_since, units = "days"))  
#calculating overall review score  
DF$overall\_review\_score=(DF$review\_scores\_rating+DF$review\_scores\_accuracy+DF$review\_scores\_cleanliness+  
DF$review\_scores\_checkin+DF$review\_scores\_communication+DF$review\_scores\_location+DF$review\_scores\_value)/7  
#calculating count of amenities  
all\_amenities = gsub("[\\[\\]\"]", "", DF$amenities)  
split\_amenities = strsplit(all\_amenities, ", ")  
flat\_amenities = unlist(split\_amenities)  
amenity\_counts = table(flat\_amenities)  
DF$amenities\_count = sapply(split\_amenities, function(x) sum(x %in% names(amenity\_counts)))  
  
#calculating score for availability\_365  
DF$availability\_365\_score = ifelse(DF$availability\_365 < 101, 4,   
 ifelse(DF$availability\_365 >= 101 & DF$availability\_365 < 201, 3,   
 ifelse(DF$availability\_365 >= 201 & DF$availability\_365 < 301, 2, 1)))  
#calculating the target variable  
DF$value <- ifelse(DF$host\_experience > 1095, 2, 1) +  
 ifelse(DF$host\_acceptance\_rate > 90, 2, 1) +  
 ifelse(DF$amenities\_count > 20, 2, 1) +  
 ifelse(DF$room\_type %in% c("Entire home/apt", "Private room"), 2, 1) +  
 ifelse(DF$availability\_365\_score > 0, DF$availability\_365\_score, 0) +  
 ifelse(DF$number\_of\_reviews\_ltm > 10, 2, 1) +  
 ifelse(DF$host\_is\_superhost == "t", 2, 1)  
DF$GoodBad=ifelse(DF$value>12,"Good","Bad")  
  
#Dropping unused features for readability  
DF <- subset(DF, select = -c(last\_scraped,amenities,minimum\_minimum\_nights,maximum\_minimum\_nights,minimum\_maximum\_nights,maximum\_maximum\_nights,minimum\_nights\_avg\_ntm,maximum\_nights\_avg\_ntm,availability\_30,availability\_60,availability\_90,availability\_365,review\_scores\_rating,review\_scores\_accuracy,review\_scores\_cleanliness,review\_scores\_checkin,review\_scores\_communication,review\_scores\_location,review\_scores\_value,calculated\_host\_listings\_count\_entire\_homes,calculated\_host\_listings\_count\_private\_rooms,calculated\_host\_listings\_count\_shared\_rooms))  
  
DF$GoodBad=as.factor(DF$GoodBad)  
#Final number of rows and columns  
str(DF)

## 'data.frame': 45382 obs. of 25 variables:  
## $ id : num 93015 13913 15400 93734 17402 ...  
## $ host\_id : int 499704 54730 60302 497514 67564 502496 388516 448154 133271 497537 ...  
## $ host\_since : Date, format: "2011-04-11" "2009-11-16" ...  
## $ host\_response\_rate : num 100 100 100 90 100 100 100 100 100 10 ...  
## $ host\_acceptance\_rate : num 25 88 41 75 100 80 97 80 96 0 ...  
## $ host\_is\_superhost : Factor w/ 2 levels "f","t": 1 1 1 2 2 2 2 1 1 1 ...  
## $ host\_identity\_verified : Factor w/ 2 levels "f","t": 2 2 2 2 2 2 2 2 2 2 ...  
## $ neighbourhood\_cleansed : chr "Hammersmith and Fulham" "Islington" "Kensington and Chelsea" "Greenwich" ...  
## $ property\_type : chr "Entire rental unit" "Private room in rental unit" "Entire rental unit" "Private room in condo" ...  
## $ room\_type : chr "Entire home/apt" "Private room" "Entire home/apt" "Private room" ...  
## $ accommodates : int 5 1 2 2 6 4 4 2 4 2 ...  
## $ price : num 175 79 150 46 476 371 120 50 151 35 ...  
## $ minimum\_nights : int 5 1 7 4 3 5 3 3 2 2 ...  
## $ maximum\_nights : int 240 29 30 365 365 365 90 180 1125 10 ...  
## $ has\_availability : Factor w/ 2 levels "f","t": 2 2 2 2 2 2 2 2 2 2 ...  
## $ number\_of\_reviews\_ltm : int 2 11 5 25 4 3 0 4 2 0 ...  
## $ instant\_bookable : Factor w/ 2 levels "f","t": 1 1 1 1 1 1 1 1 2 1 ...  
## $ calculated\_host\_listings\_count: int 1 2 1 1 9 1 1 2 11 1 ...  
## $ reviews\_per\_month : num 0.27 0.26 0.56 1.21 0.36 0.16 0.76 0.53 0.74 0.22 ...  
## $ host\_experience : num 4531 5042 5023 4533 4993 ...  
## $ overall\_review\_score : num 4.79 4.74 4.84 4.67 4.74 ...  
## $ amenities\_count : int 48 55 25 46 38 54 11 49 27 15 ...  
## $ availability\_365\_score : num 4 1 4 3 2 2 4 2 4 1 ...  
## $ value : num 13 11 13 14 13 12 14 11 14 9 ...  
## $ GoodBad : Factor w/ 2 levels "Bad","Good": 2 1 2 2 2 1 2 1 2 1 ...

**Model Building & Model Evaluation**

**Decision Tree Algorithm**

* Decision tree is used for predicting the target variable. They work like how humans make decisions. It involves a series of if-then splits called binary recursion splitting. Considering all the features, split is made at the feature where the decision tree can reduce most errors.
* Decision Trees are easy to understand and interpret. The model’s decision-making process is represented as a tree structure with clear rules at each node, making it accessible to non-experts.
* Airbnb data often includes a mix of categorical and numerical features, such as room type, amenities, and pricing. Decision Trees can handle all types of features without requiring extensive preprocessing.
* Decision trees do not assume a specific distribution of the data, being a non-distance-based algorithm, we can work with the raw, unscaled features.
* Decision trees can model non-linear relationships between features, making them suitable for complex data patterns.
* For all the above advantages, Decision tree algorithm is used in predicting the target variable for Airbnb dataset.
* The code below uses k-fold cross-validation using the caret and rpart packages in R to assess the performance of a decision tree model on the given dataset. The decision tree is trained on different subsets of the data, and the average accuracy is calculated as a measure of model performance. The decision tree is visualized using rpart.plot.
* We can observe that the below code produces an accuracy of 62%. This depends on the features selected for building the model and splitting of the training and testing data. Techniques like k-fold cross validation, trains the model on multiple splits to give better accuracy.This could sometimes lead to overfitting when the tree is deep and try to capture noise in the training data. under such cases, the model may not generalize well for new, unseen data. The prediction will vary significantly for small variation resulting in prediction errors.

install.packages("caret")

install.packages("rpart.plot")

install.packages("rpart")

library(rpart)  
library(tree)  
  
DF1=DF[,c(4,7,11,12,15,17,18,19,25)]  
num\_folds=10  
library(caret)

# create the folds

folds=createFolds(DF1$GoodBad,k=num\_folds,list=TRUE,returnTrain=FALSE)

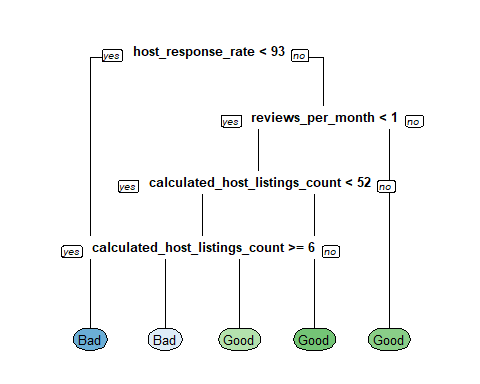
# create variable cv\_result for storing the accuracy of different folds  
cv\_result=numeric(length(folds))  
  
for(i in 1:num\_folds){

# create test and train data  
 train\_data <- DF1[-folds[[i]], ]  
 test\_data <- DF1[folds[[i]], ]

# model creation  
 DTmodel=rpart(GoodBad~.,train\_data,method='class')

# accuracy prediction  
 predictions <- predict(DTmodel, newdata = test\_data,type='class')  
 confusion\_matrix=table(predictions,test\_data$GoodBad)  
 accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)  
 cv\_result[i] <- accuracy  
}

# plotting the decision tree  
library(rpart.plot)  
rpart.plot(DTmodel, yesno = 2, type = 0, extra = 0, cex = 0.8)



average\_accuracy <- mean(cv\_result)  
print(average\_accuracy)

# Accuracy of decision tree after k-fold cross validation

## [1] 0.6237716

**Pruning**

Pruning is a technique used in decision trees to prevent overfitting and to improve generalization of the model. It tries to strike a balance between complexity and accuracy. In this technique branches of those feature which do not contribute significantly to the model’s predictive accuracy are removed.

In pruning, the accuracy of the model will be reduced to perform better on generalization, prediction, and to reduce complexity. In the below code, prune(DTmodel2, cp = 0.03) is used to prune the model. cp is the complexity parameter which can be determined by plotting, plotcp(DTmodel2). This shows the value of cp and relative error. we can select the appropriate value of cp to reduce the errors. The output tree after pruning will have less number of branches and nodes generalizing them to new data.

The accuracy of the model after pruning reduced from 62% to 60%

install.packages("caret")

install.packages("rpart.plot")

install.packages("rpart")

library(rpart)  
library(tree)  
  
DF1=DF[,c(4,7,11,12,15,17,18,19,25)]  
num\_folds=10  
library(caret)

# create the folds

folds=createFolds(DF1$GoodBad,k=num\_folds,list=TRUE,returnTrain=FALSE)

# create variable cv\_result for storing the accuracy of different folds  
cv\_result=numeric(length(folds))

for(i in 1:num\_folds){

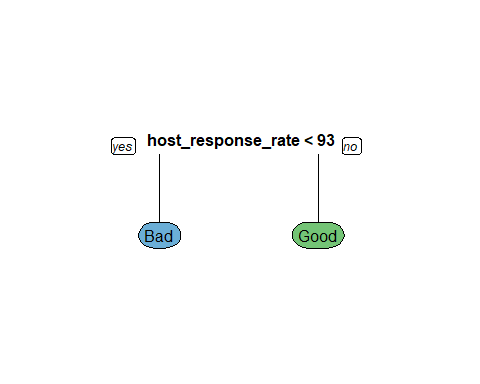
# create test and train data  
 train\_data <- DF1[-folds[[i]], ]  
 test\_data <- DF1[folds[[i]], ]

# model creation  
DTmodel2=rpart(GoodBad~.,train\_data,method='class')  
# pruned model creation  
pruned\_rpart\_model <- prune(DTmodel2, cp = 0.03)

# accuracy prediction  
 predictions <- predict(pruned\_rpart\_model, newdata = test\_data,type='class')  
 confusion\_matrix=table(predictions,test\_data$GoodBad)  
 accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)  
 cv\_result[i] <- accuracy  
}

# plotting the pruned decision tree

library(rpart.plot)  
rpart.plot(pruned\_rpart\_model,yesno=2,type=0,extra=0)



average\_accuracy <- mean(cv\_result)  
print(average\_accuracy)

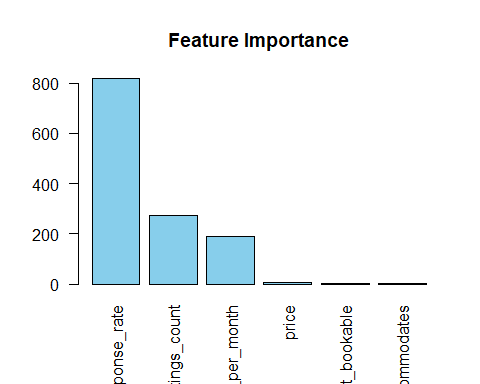
# Accuracy of pruned decision tree after k-fold cross validation

## [1] 0.6080823

**Feature Importance**

DTmodel=rpart(GoodBad~.,train\_data,method='class')  
feature\_importances <- DTmodel$variable.importance  
print(feature\_importances)

## host\_response\_rate calculated\_host\_listings\_count   
## 817.1158319 271.2771496   
## reviews\_per\_month price   
## 190.0412056 7.0272428   
## instant\_bookable accommodates   
## 1.5962456 0.8289619

feature\_names <- names(feature\_importances)  
barplot(feature\_importances, names.arg = feature\_names, col = "skyblue", main = "Feature Importance",las = 2)

- Feature\_importance shows the importance scores for each feature in descending order. The values indicate the contribution of each feature to the model’s decision-making process.

- We can use this information to understand which features are more influential in predicting the target variable (GoodBad).

- From the above barchart, we can observe that features like host\_response\_rate, calculated\_host\_listings\_count, review\_per\_month and price appears to be more significant according to their higher importance scores.

**Discussion**

**Insights**

- Higher response rate positively contributes to the target variables 'GoodBad' indicating that hosts who promptly responds to the inquiries offer good experience to the customers. Airbnb should encourage hosts to have high response rate to maintain in good listing and attract customers.

- Host having many numbers of listings might be highly experienced and reliable. To attract customers for listings, the host should manage multiple listings efficiently and consider the quality of each listing.

- High number of reviews contribute positively to the target value. Hosts should maintain high guest engagement and provide excellent services and amenities.

- price plays an important role in determining the value of the listing. Host should carefully consider their pricing strategy based on the market trends, competitor price and based on the facilities provided.

**Crucial results**

- The most crucial result was the creation of the variable 'GoodBad'. This new variable represents a binary classification. Conditions are applied to various features such as host\_experience, host\_acceptance\_rate, amenities\_count, room\_type, availability\_365\_score, number\_of\_reviews\_ltm, and host\_is\_superhost to arrive at this value. The logic behind these conditions is to create a score (DF$value) by assigning different weights to the features. If DF$value is greater than the threshold 12, it is considered good listing otherwise it is labeled as bad listing. Domain knowledge is crucial for drafting conditions to capture underlying pattern in the data effectively.

-Another crucial result is the accuracy of decision tree before and after pruning. Pruning is a technique used to reduce overfitting and improve generalization for the model. In this analysis, there was a reduction in accuracy from 62% to 60% after pruning. This drop in accuracy is due to certain branches being removed from the decision tree. This change in accuracy is crucial as it provides an insight into trade-off between model complexity and generalization.

-The proposed algorithm showed an accuracy of 60% and not higher values because the features used in calculating the target variables were not included in the training data. This will prevent the model from overfitting, showing high accuracy and misleading insights.

- To address these limitations, sufficient domain knowledge should be gathered to identify and include features that provide correct predictions.

**Limitations and improvements**

- Overfitting: Decision tree can be overfitting capturing noise in the training data and not fitting well for the unseen data. This can be overcome by using pruning techniques.

-They may not handle missing data. Imputation methods can be employed to address missing data issues.

- Small change in data can result in completely different tree structure, making decision tree unstable. Techniques such as random forest which combines multiple trees can be used to improve stability.

-Decision tree may not perform well when there are multiple levels of categorical variables. Techniques like one-hot encoding can be employed to handle categorical variables with multiple levels.