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

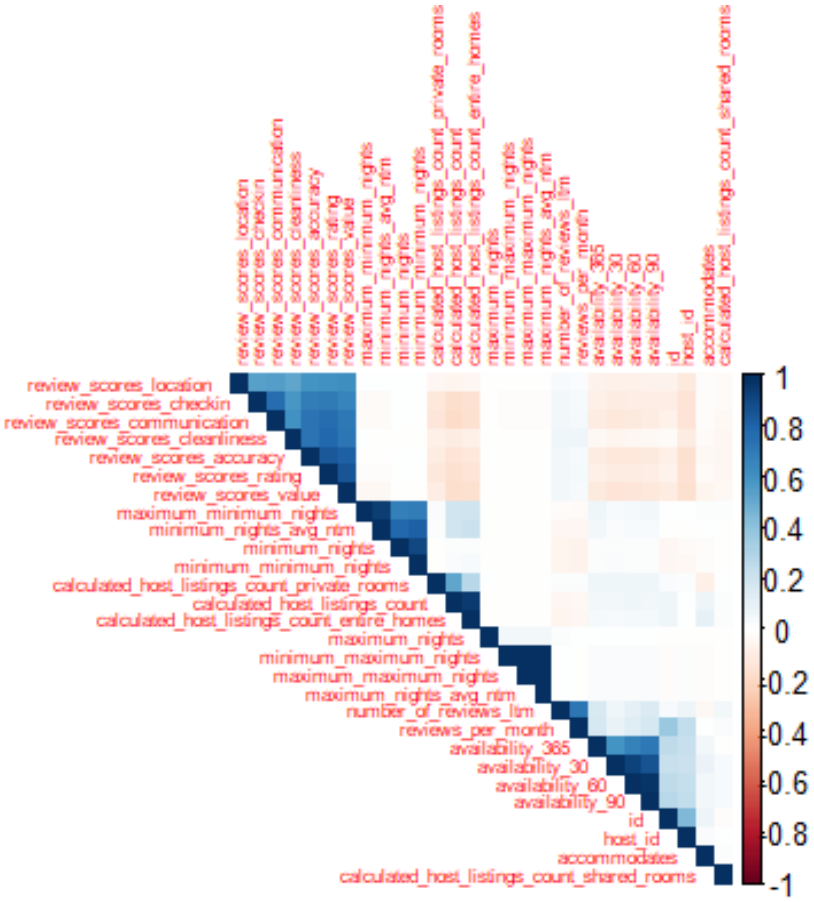
This includes loading the data, understanding its structure and steps used in cleaning the data.

```
# 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 ,n
eighborhood_overview,picture_url,host_url,host_name,host_location,host_about,
host_thumbnail_url,host_picture_url,host_neighbourhood,host_listings_count,ho
st_total_listings_count,host_verifications,host_has_profile_pic,host_response
_time,neighbourhood,neighbourhood_group_cleansed,bathrooms,bathrooms_text,bed
rooms,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_nig
hts,minimum_maximum_nights,maximum_maximum_nights,minimum_nights_avg_ntm,maxi
mum_nights_avg_ntm,review_scores_rating,review_scores_accuracy,review_scores_
cleanliness,review_scores_checkin,review_scores_communication,review_scores_l
ocation,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.c
ex = 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

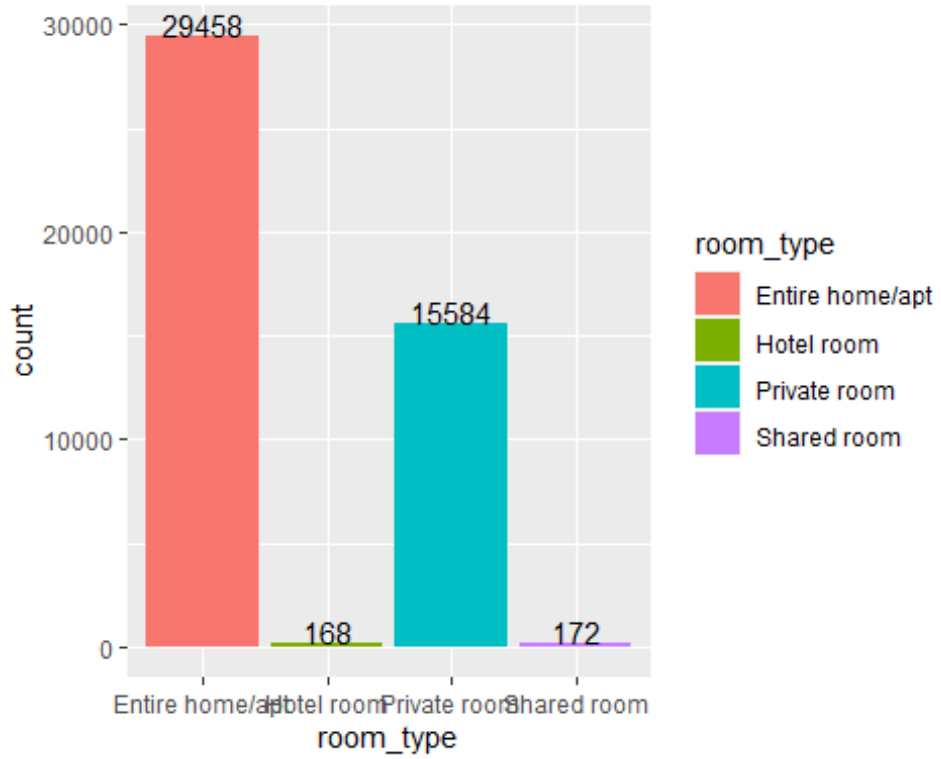
-Converting datatypes of features to Date, factor, removing $ and % symbol from numbers and converting them to numeric datatype.

-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")
```



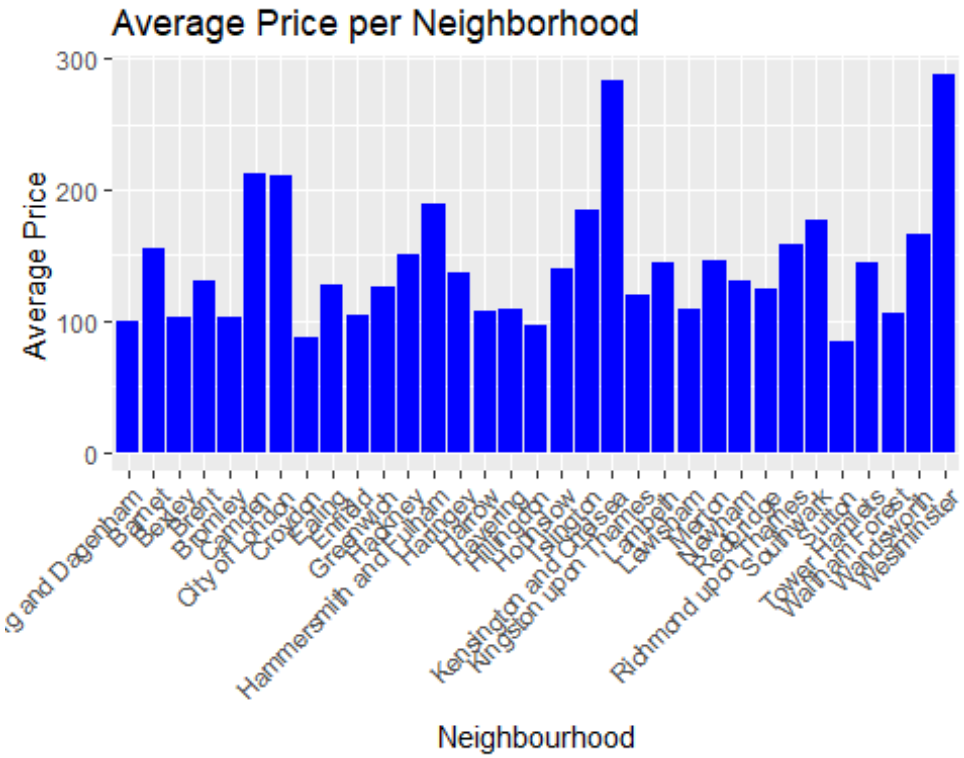

```
  labs(title = "Number of Listings by Room Type",
       x = "Room Type",
       y = "Number of Listings")
```

```
## $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$host_since, unit
s = "days"))
#calculating overall review score
DF$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_loca
tion+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(ame
nity_counts)))

#calculating score for availability_365
DF$availability_365_score = ifelse(DF$availability_365 < 101, 4,
                            ifelse(DF$availability_365 >= 101 & DF$availabili
ty_365 < 201, 3,
                            ifelse(DF$availability_365 >= 201 & DF$availabili
ty_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,ma
ximum_minimum_nights,minimum_maximum_nights,maximum_maximum_nights,minimum_ni
ghts_avg_ntm,maximum_nights_avg_ntm,availability_30,availability_60,availabil
ity_90,availability_365,review_scores_rating,review_scores_accuracy,review_sc
ores_cleanliness,review_scores_checkin,review_scores_communication,review_sco
res_location,review_scores_value,calculated_host_listings_count_entire_homes,
calculated_host_listings_count_private_rooms,calculated_host_listings_count_s
hared_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 50
2496 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" "Islingto
n" "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 3
5 ...
##  $ 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

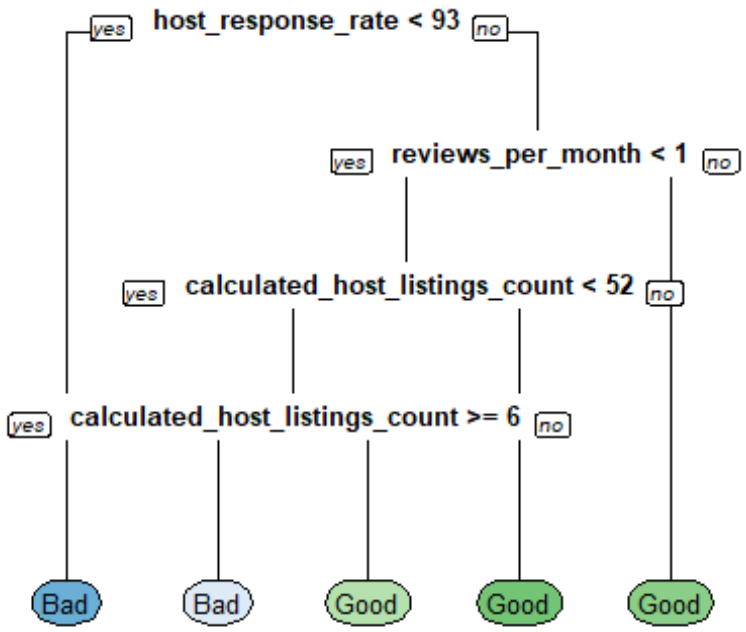
Decision Tree Algorithm

- 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
```

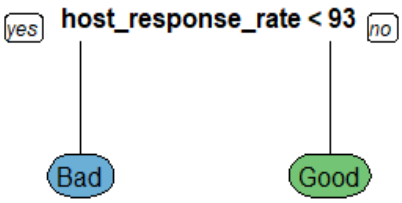
Pruning

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)
```

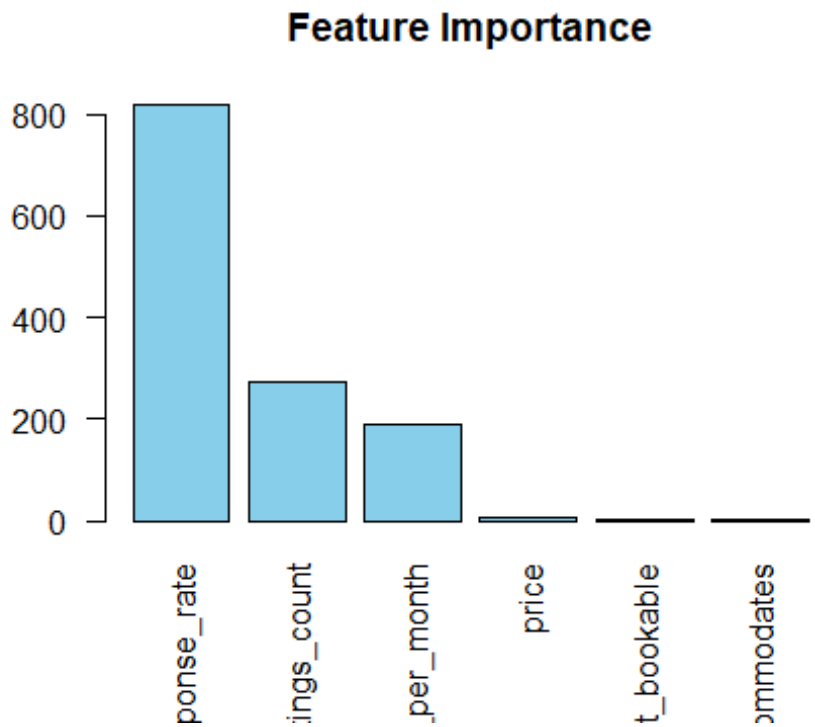
```
# Accuracy of pruned decision tree after k-fold cross validation

## [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
```

```r
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.