# Load necessary libraries  
library(dplyr)  
library(randomForest)  
library(caret)  
library(ggplot2)  
  
# Load the dataset  
hotel\_bookings <- read.csv("hotel\_bookings.csv")  
  
# Dropping columns  
hotel\_bookings <- hotel\_bookings %>%  
 select(-company, -arrival\_date\_week\_number)  
  
# Handling missing values  
# Display columns with missing values  
# Check for "NULL" values in each column  
print(colSums(hotel\_bookings == "NULL"))  
  
# Replace "NULL" values with median for 'children' column  
hotel\_bookings$children <- ifelse(hotel\_bookings$children == "NULL", median(as.numeric(hotel\_bookings$children), na.rm = TRUE), hotel\_bookings$children)  
  
# Replace "NULL" values with median for 'agent' column  
hotel\_bookings$agent <- ifelse(hotel\_bookings$agent == "NULL", median(as.numeric(hotel\_bookings$agent), na.rm = TRUE), hotel\_bookings$agent)  
  
# Find the most frequent value in the 'country' column  
most\_frequent\_country <- names(sort(table(hotel\_bookings$country), decreasing = TRUE))[1]  
  
# Replace "NULL" values with the most frequent value in the 'country' column  
hotel\_bookings$country <- ifelse(hotel\_bookings$country == "NULL", most\_frequent\_country, hotel\_bookings$country)  
  
# Check for remaining missing values  
print(colSums(hotel\_bookings == "NULL"))  
  
print(str(hotel\_bookings))  
  
  
# Convert 'arrival\_date\_year' to character  
hotel\_bookings$arrival\_date\_year <- as.character(hotel\_bookings$arrival\_date\_year)  
  
# Checking for miswritings  
# Display unique values for selected columns  
selected\_cols <- c("hotel", "arrival\_date\_month", "arrival\_date\_year", "country", "market\_segment", "distribution\_channel", "reserved\_room\_type", "assigned\_room\_type", "deposit\_type", "customer\_type", "reservation\_status")  
unique\_values <- lapply(hotel\_bookings[selected\_cols], unique)  
print(unique\_values)  
  
# Check for logical errors  
logical\_errors <- hotel\_bookings %>%  
 filter(children == 0 & adults == 0 & babies == 0)  
  
# Display rows with logical errors  
print(logical\_errors)  
.   
# Drop rows with logical errors  
hotel\_bookings <- hotel\_bookings %>%  
 filter(!(children == 0 & adults == 0 & babies == 0)) %>%  
 arrange()  
  
# Reset index  
hotel\_bookings <- hotel\_bookings %>%  
 add\_rownames("index")  
  
# Check the cleaned dataset  
print(str(hotel\_bookings))  
  
# Combine and format as character with error handling  
hotel\_bookings$arrival\_date <- sapply(1:nrow(hotel\_bookings), function(i) {  
 year <- hotel\_bookings$arrival\_date\_year[i]  
 month <- match(hotel\_bookings$arrival\_date\_month[i], month.name)  
 day <- hotel\_bookings$arrival\_date\_day\_of\_month[i]  
   
 # Check for missing values and return NA if any  
 if (any(is.na(c(year, month, day)))) {  
 return(NA)  
 }  
   
 # Combine the values with the desired format  
 paste0(year, "-", sprintf("%02d", month), "-", sprintf("%02d", day))  
})  
  
# Drop the individual columns  
hotel\_bookings <- hotel\_bookings %>%  
 select(-c(arrival\_date\_year, arrival\_date\_month, arrival\_date\_day\_of\_month))  
  
  
summary(hotel\_bookings)  
  
  
# Identify outliers in lead\_time  
lead\_time\_outliers <- boxplot.stats(hotel\_bookings$lead\_time)$out  
print(lead\_time\_outliers)  
upper\_whisker <- boxplot.stats(hotel\_bookings$lead\_time)$stats[5]  
hotel\_bookings$lead\_time[hotel\_bookings$lead\_time > upper\_whisker] <- upper\_whisker  
  
# Identify outliers in stays\_in\_weekend\_nights and stays\_in\_week\_nights  
weekend\_nights\_outliers <- boxplot.stats(hotel\_bookings$stays\_in\_weekend\_nights)$out  
week\_nights\_outliers <- boxplot.stats(hotel\_bookings$stays\_in\_week\_nights)$out  
print(week\_nights\_outliers)  
print(weekend\_nights\_outliers)  
# Replace outliers with a reasonable value (e.g., the upper bound of a whisker)  
upper\_whisker\_weekend <- boxplot.stats(hotel\_bookings$stays\_in\_weekend\_nights)$stats[5]  
upper\_whisker\_week <- boxplot.stats(hotel\_bookings$stays\_in\_week\_nights)$stats[5]  
  
hotel\_bookings$stays\_in\_weekend\_nights[hotel\_bookings$stays\_in\_weekend\_nights > upper\_whisker\_weekend] <- upper\_whisker\_weekend  
hotel\_bookings$stays\_in\_week\_nights[hotel\_bookings$stays\_in\_week\_nights > upper\_whisker\_week] <- upper\_whisker\_week  
  
# Identify outliers in adults, children, babies  
adults\_outliers <- boxplot.stats(hotel\_bookings$adults)$out  
upper\_whisker\_adults <- boxplot.stats(hotel\_bookings$adults)$stats[5]  
hotel\_bookings$adults[hotel\_bookings$adults > upper\_whisker\_adults] <- upper\_whisker\_adults  
babies\_outliers <- boxplot.stats(hotel\_bookings$babies)$out  
print(babies\_outliers)  
upper\_whisker\_babies <- boxplot.stats(hotel\_bookings$babies)$stats[5]  
hotel\_bookings$babies[hotel\_bookings$babies %in% c(9, 10)] <- upper\_whisker\_babies  
  
# Identify outliers in children  
children\_outliers <- boxplot.stats(hotel\_bookings$children)$out  
upper\_whisker\_children <- boxplot.stats(hotel\_bookings$children)$stats[5]  
hotel\_bookings$children[hotel\_bookings$children == 10] <- upper\_whisker\_children  
print(children\_outliers)  
  
# Find the minimum ADR value  
min\_adr <- min(hotel\_bookings$adr)  
  
# Find the median ADR value  
median\_adr <- median(hotel\_bookings$adr, na.rm = TRUE)  
  
# Replace the minimum ADR value with the median ADR value  
hotel\_bookings$adr[hotel\_bookings$adr == min\_adr] <- median\_adr  
  
#visualization#  
  
# Bar chart for hotel distribution  
ggplot(hotel\_bookings, aes(x = hotel, fill = hotel)) +  
 geom\_bar() +  
 labs(title = "Distribution of Hotel Types", x = "Hotel Type", y = "Count") +  
 theme\_minimal()  
  
# Bar plot for market segmentation  
ggplot(hotel\_bookings, aes(x = market\_segment, fill = hotel)) +  
 geom\_bar(position = "dodge", alpha = 0.7) +  
 labs(title = "Distribution of Bookings by Market Segment", x = "Market Segment", y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
# Stacked bar plot for reservation status  
ggplot(hotel\_bookings, aes(x = reservation\_status, fill = hotel)) +  
 geom\_bar(position = "stack", alpha = 0.7) +  
 labs(title = "Proportion of Reservation Status", x = "Reservation Status", y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
# Convert 'arrival\_date' to Date format  
hotel\_bookings$arrival\_date <- as.Date(hotel\_bookings$arrival\_date)  
  
# Count the number of bookings for each date  
booking\_counts <- hotel\_bookings %>%  
 group\_by(arrival\_date) %>%  
 summarise(booking\_count = n())  
  
# Line Chart for Number of Bookings Over Time  
line\_chart\_bookings\_over\_time <- ggplot(booking\_counts, aes(x = arrival\_date, y = booking\_count)) +  
 geom\_line() +  
 labs(title = "Line Chart: Number of Bookings Over Time",  
 x = "Arrival Date",  
 y = "Number of Bookings")  
  
print(line\_chart\_bookings\_over\_time)  
  
##Modelling  
  
# Convert 'is\_canceled' to a factor  
hotel\_bookings$is\_canceled <- as.factor(hotel\_bookings$is\_canceled)  
  
# Split the data into training and testing sets  
set.seed(123)  
train\_indices <- createDataPartition(hotel\_bookings$is\_canceled, p = 0.8, list = FALSE)  
train\_data <- hotel\_bookings[train\_indices, ]  
test\_data <- hotel\_bookings[-train\_indices, ]  
  
# Create a random forest model  
# Impute missing values with the median for numeric variables  
train\_data <- train\_data %>%  
 mutate\_if(is.numeric, funs(ifelse(is.na(.), median(., na.rm = TRUE), .)))  
  
# Create a random forest model  
rf\_model <- randomForest(is\_canceled ~ ., data = train\_data, ntree = 100)  
  
# Make predictions on the test set  
predictions <- predict(rf\_model, newdata = test\_data)  
  
# Evaluate the model  
confusion\_matrix <- table(predictions, test\_data$is\_canceled)  
accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)  
print(paste("Accuracy:", round(accuracy, 4)))  
  
# You can also check other metrics like precision, recall, and F1-score if needed  
precision <- confusion\_matrix[2, 2] / sum(confusion\_matrix[, 2])  
recall <- confusion\_matrix[2, 2] / sum(confusion\_matrix[2, ])  
f1\_score <- 2 \* (precision \* recall) / (precision + recall)  
  
print(paste("Precision:", round(precision, 4)))  
print(paste("Recall:", round(recall, 4)))  
print(paste("F1 Score:", round(f1\_score, 4)))  
  
# Display summary of the cleaned dataset  
summary(hotel\_bookings)