An Analysis of Hotel Booking Cancellations

BAIS:6110

April 30, 2023

Prepared by: Ifeoma Anyaoha, Matteo Hernandez, Evan Parker, Somer Sinnard, Vikas Yadav

Table of Contents: **Page**

I. Dataset and Motivating Questions

The dataset used for this project was obtained from the Kaggle website and is composed of hotel booking information for two hotels, a city hotel and resort hotel. Each of the 119,390 records represents one hotel reservation and descriptive data regarding that reservation is recorded in the thirty-two columns available in the data. A list of those variables can be seen in the appendix under *Item A***.**

Our team had several goals in mind when completing our analysis. We first aimed to describe the data at hand and identify the types of guests that had made reservations with the two hotels, in the context of cancellations. As a manager at one of these hotels, it would be useful to understand the demographics canceling bookings and what kinds of bookings are most likely canceled. By focusing on how our different descriptive variables aligned with the cancellations, we can draw conclusions about what factors may lead to canceled reservations. The following questions guided our exploratory analysis:

A. *Are there more bookings at the city or resort hotel? Do the percentages of cancellations differ?*
B. *How far in advance do guests typically book their stay? Does that time frame have any bearing on the cancellation rate?*
C. *What are the busiest months for bookings? Do those months also have the highest cancellation rates?*
D. *What kind of guest groupings result in the most total bookings? Do any of these groups cancel more often than others?*
E. *Which meal types and room types are most popular among guests that end up canceling?*
F. *How many cancellations come from returning guests?*
G. *Does the market segment a booking belongs to have an impact on whether the stay will be cancelled?*
H. *Does the country of origin of the guest have any impact on whether a booking will be canceled?*
I. *We expect the length of a stay to be strongly correlated with the cost of the stay, is that accurate?*
J. *What cost range do the average daily rates fall in for these hotels?*

With a similar goal, we utilized classification methods to build a predictive model centered around predicting a booking being canceled. With the model, we could input the description of the booking and obtain the likelihood of whether a guest will complete the reservation or cancel, which could serve as a useful tool for managing bookings.

Modeling the several factors that go into the price of a stay is also important. Using regression analysis, we produced a model that allows for the calculation of the cost of a one-night stay in May, the busiest month on average for these hotels, given all the variables available in the data. This was also created in mind as a tool for management to use and discern what cost changes, if any, should be made to maximize revenue during that profitable period.

Most of this report focuses on assessing cancellation data as it is an area where we believe most of the business opportunity is unidentified. Crafting a profitable analysis that could be used by a manager or hotel brand to produce action points, lower cancellations, and produce more revenue was our overarching goal.

II. Loading Data into Hive Tables

Our team used a SQL notebook as our default language and included %r in cells utilizing R code. To load in the data for analysis, the CSV file was first copied over from the FileStore tables into the user's data:

```
%fs cp "/FileStore/tables/hotel_bookings.csv" "/users/vikyadav/hotel_bookings.csv"
```

Hive SQL code was then used to create a Hive table named *hotel_bookings* to contain the data in the *hotel_bookings.csv* file. Code was included to drop any pre-existing tables named hotel_bookings before creating the new table and the code specified that the first row in the data was a header row. This portion of the code is lengthy and can be found in the appendix under *Item B.* The following code returned the newly created table and verified that reading in the data was successful:

```
select * from hotel_bookings;
```

While our team now had a Hive table that could be used for analysis, partitioning of the table was required to increase the output speed for code appearing later in our analysis. To allow for dynamic partitioning in our notebook the following code was run:

```
SET hive.exec.dynamic.partition = true;

SET hive.exec.dynamic.partition.mode = nonstrict;

SET hive.exec.max.dynamic.partitions.pernode = 1500;
```

Partitioning was then completed to break down the bookings data by year of reservation and month of reservation to assist in executing code that required an exceptionally high volume of data. This partitioning split the data from the existing *hotel_bookings table* into chunks based on two columns detailing the reservation date information. The code used to create the new partitioned table named *hotel_bookings_part* is found in the appendix under *Item C.* The code used to insert the data from *hotel_bookings* into the new table according to the specified schema is found below:

```
INSERT INTO TABLE hotel_bookings_part partition(res_year, res_month)

SELECT *, arrival_date_year as res_year, arrival_date_month as res_month

FROM hotel_bookings;
```

The new partitioned table was verified by running the two lines of code below and data insertion was found to be successful:

```
SELECT * from hotel_bookings_part limit 10;

show PARTITIONS hotel_bookings_part;
```

III. Data Summarization

The descriptive information for each booking record was examined alongside the cancellations data to gain a better understanding of the dataset and identify areas with high cancellation rates. Our motivating questions were addressed using both SQL and R code to extract relevant data and create visualizations.

While SQL code was used to query the hotel_bookings table to answer some of our questions and produce visualizations based on the appropriate outputs, it was also necessary to convert the hotel_bookings table into a Spark DataFrame to be utilized. The code for this step is found below.

```
%r

query <- "select * from hotel_bookings_part"

hotel_df <- sql(query)
```

For the visualizations and models produced that required R code to be used, an R data frame was also created. The code for this step is found below:

```
%r

df_final <-as.data.frame(hotel_df)
```

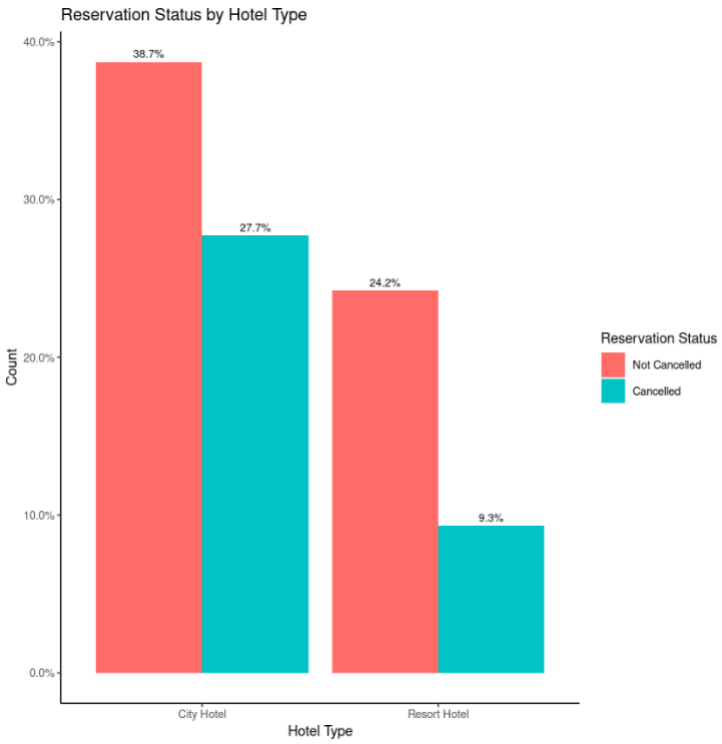
A) Are there more bookings at the city or resort hotel? Do the percentages of cancellations differ?

The R code used to obtain this visualization can be found in the appendix under Item D.

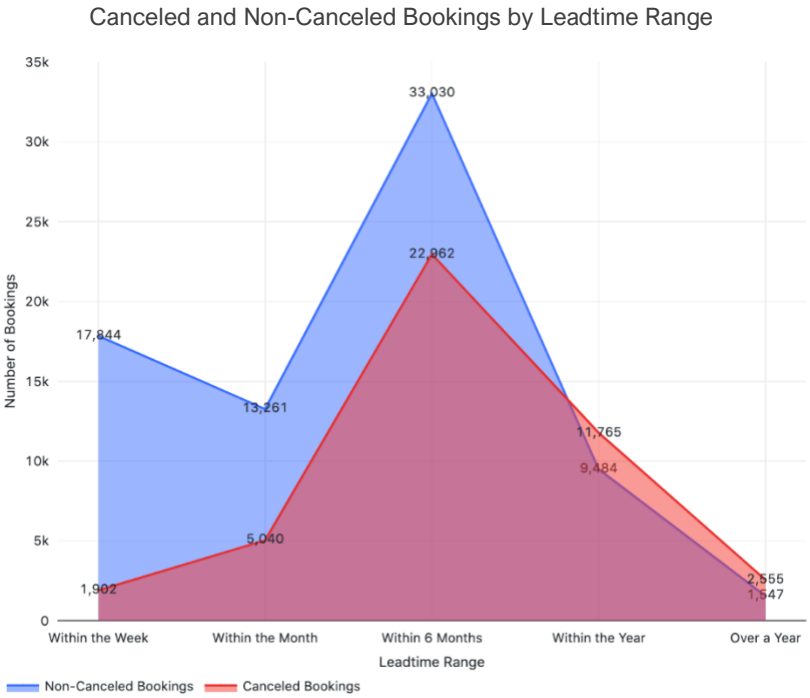
This graph depicts the cancelled and not cancelled reservations based on the two hotels in the data, which are the city hotel and the resort hotel.

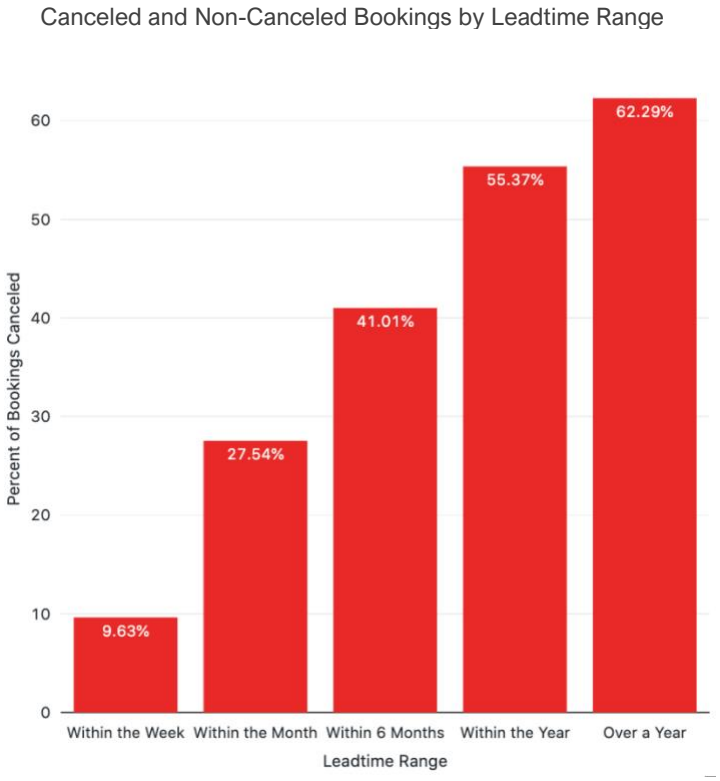
We see that reservations at the city hotel are far more numerous than those at the resort hotel. The city hotel has nearly twice the number of reservations of the resort hotel with a total of 79,330 bookings, whereas the resort hotel only had 40,060 bookings.

While the city hotel has more bookings, there are far more cancellations for this hotel than the resort hotel as well. While the resort hotel may struggle to obtain a large number of reservations, it has a far better retention rate for non-canceled bookings overall.



B) How far in advance do guests typically book their stay? Does that time frame have any bearing on the cancellation rate?





The SQL code used to obtain these visualizations can be found in the appendix under Item E.

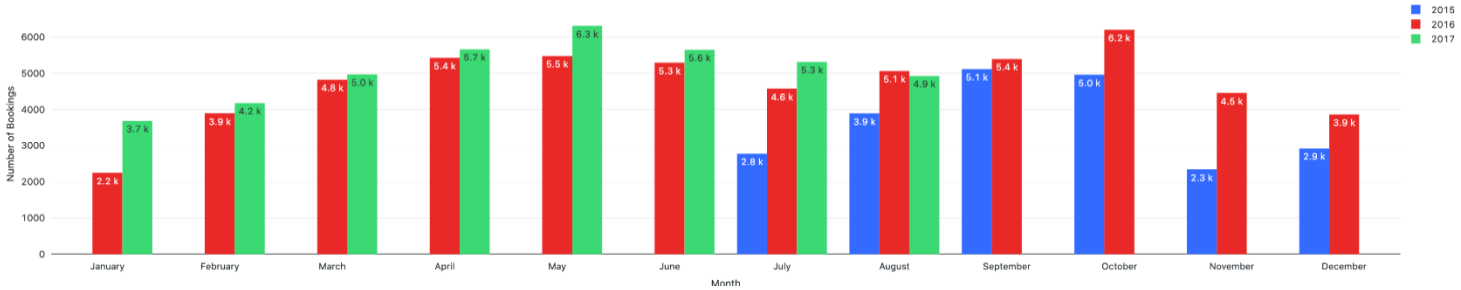
The leadtime range in which the least cancellations are made in comparison to non-cancelled bookings is within the week of the arrival date. Up to the within six months range, non-canceled bookings outnumber canceled bookings and generally follow the same pattern.

Once the booking is made six months or further out from the arrival date, it becomes more likely than not that a reservation will be canceled by the guest.

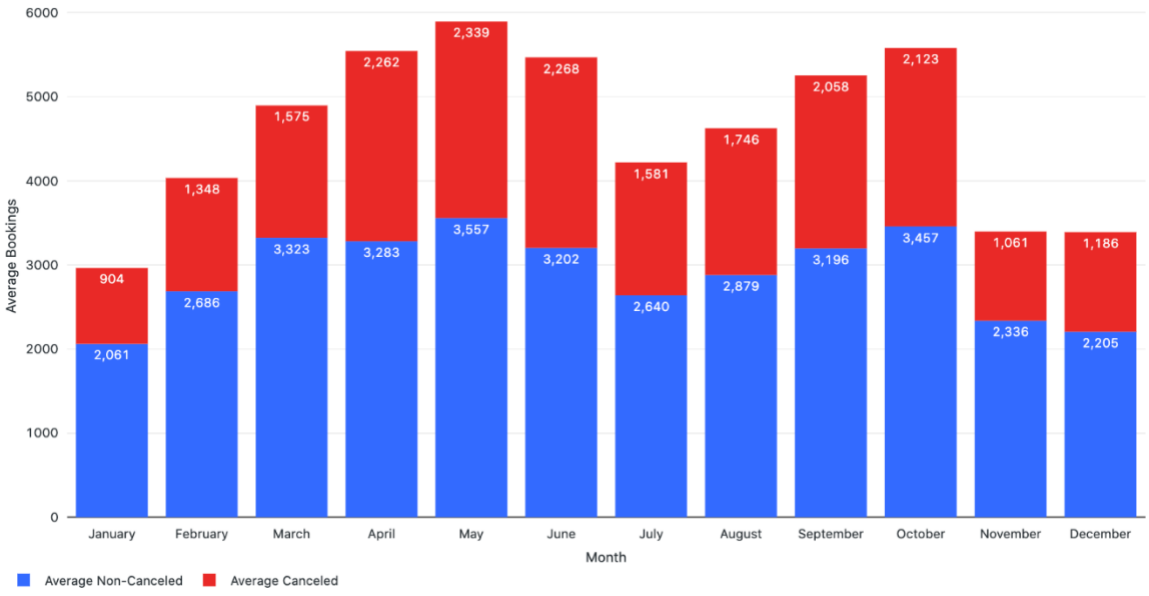
It is expected that stays that are booked far in advance would likely be subject to more cancellations as there is more opportunity for timing conflicts to arise. While this is notable, the bulk of the bookings are made within 6 months of the arrival date and high cancellation rates in those ranges would be more alarming from a revenue perspective.

C) What are the busiest months for bookings? Do those months also have the highest cancellation rates?

The dataset was comprised of booking records across three years, 2015, 2016, and 2017. Unfortunately, bookings data was not recorded for each month in each year. Below we see that in 2015 bookings were only recorded for the months from July to December. In 2016 there was bookings data available for every month. In 2017 bookings were only recorded for the months from January to August. Below the total number of bookings corresponding to each year and month is plotted. The top three months with the highest reported bookings are May of 2017, October of 2016, and May of 2016. The bottom three months with the lowest reported bookings are January of 2016, November of 2015, and July of 2015. *The SQL code used to obtain this visualization is found in the appendix under Item K.*



Due to our uneven dataset for some months, average bookings were calculated for each month so general booking trends could be observed across the 12 months without skewing the data for the months of July and August that have bookings data available in all three years. Below we see these average bookings plotted as well as the average number of cancellations for each month. We see that May is typically the strongest month for bookings for these two hotels and January is the weakest. The month with the highest cancellation rate on average is June with 41.46% of bookings cancelled and the month with the lowest cancellation rate on average is January at 30.48%. *The SQL code used to obtain this visualization is found in the appendix under Item L.*

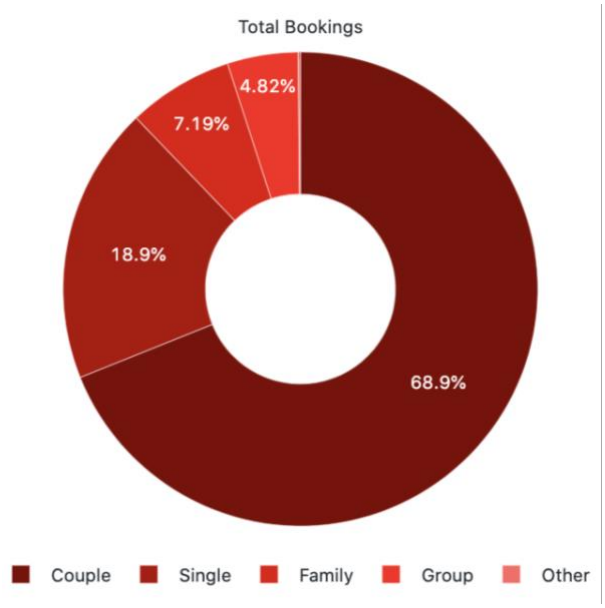


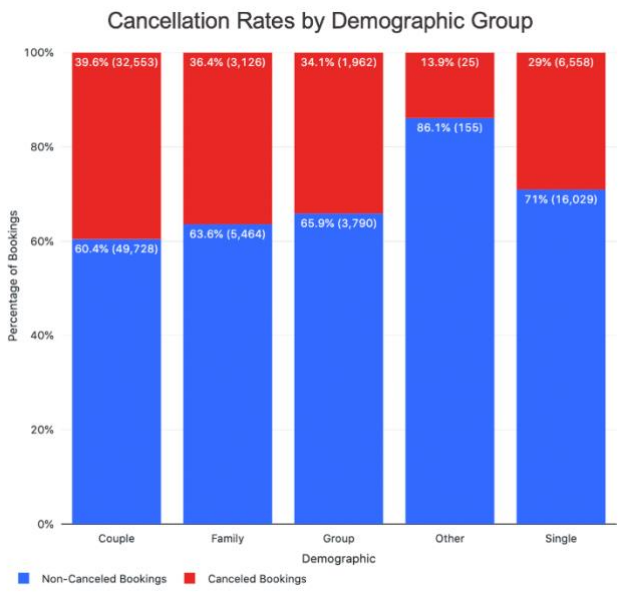
D) What kind of guest groupings result in the most total bookings? Do any of these groups cancel more often than others?

The adults, children, and babies variables were used to characterize the guest groupings found within the data. We see that couples comprise by far most of the bookings at 68.9%. Singles also make up a sizeable percentage of the bookings and we see that families, adult groups, and other types of guest groups make up a small share of the bookings recorded.

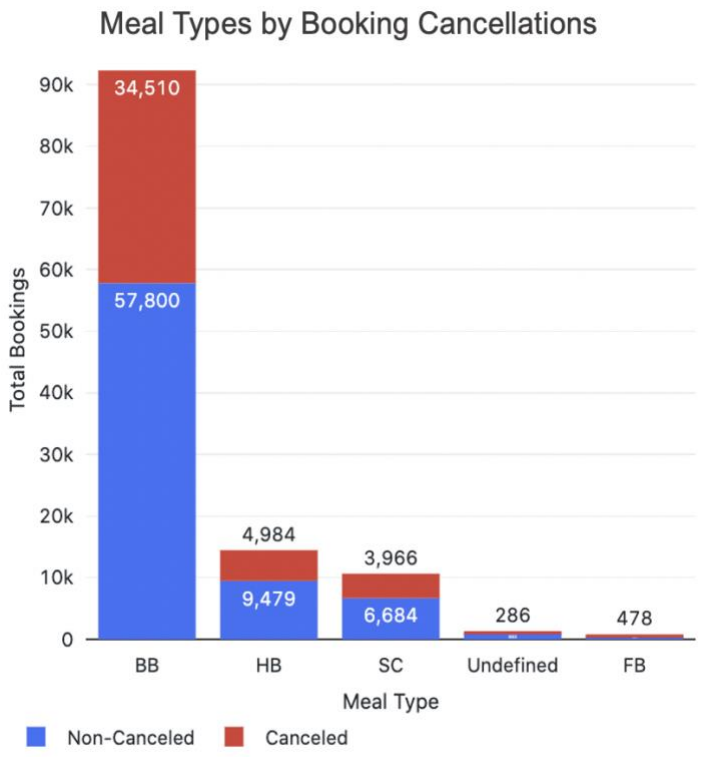
We also see that the group with the highest cancellation rate is the "Other" group. While we aren't sure what groups of guest fall into this category, we know that the group's share of bookings is very low in comparison to the others and deem the high cancellation rate not of deep concern. While the rest of the groups don't show much variation in their cancellation rates, the "Single" group has the highest cancellation rate at 71% and the "Couple" group has the lowest cancellation rate at 60.4%. These two are the most sizeable groups in terms of bookings share so these values are far more significant.

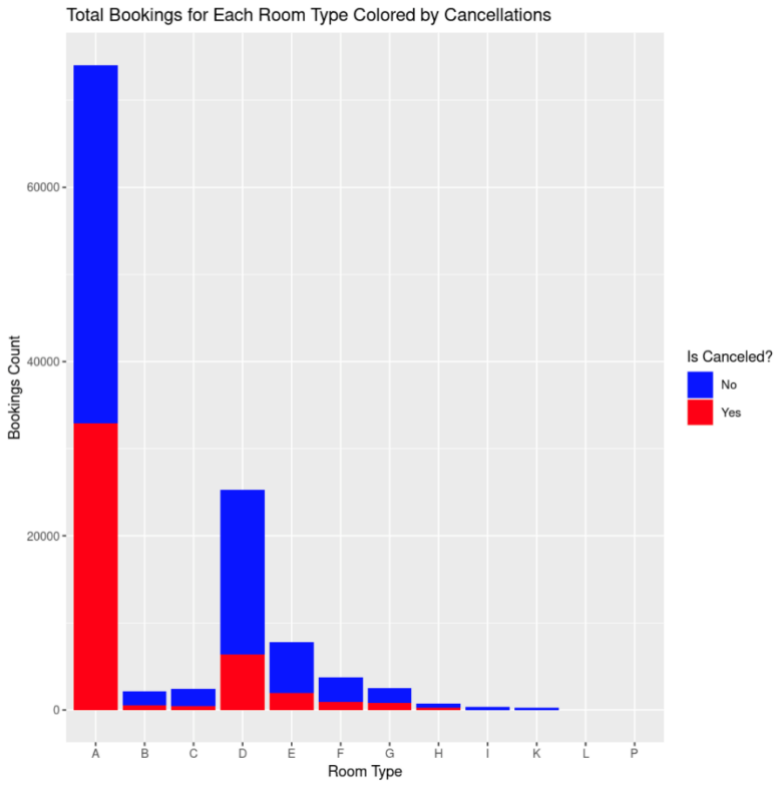
The SQL code used to obtain these visualizations is found in the appendix under Item F.





E) Which meal types and room types are most popular among guests that end up canceling?





For both variables a stacked bar chart was created to compare cancellations with overall bookings for these guest options.

The SQL code to create the meal type chart is found below:

```
SELECT meal, total, canceled, total-canceled as
non_canceled, canceled/total*100
FROM
(select meal, count(*) as total, sum(is_canceled) as
canceled
from hotel_bookings
group by meal)
order by total desc
```

The R code to create the room type chart is found in the appendix under Item G.

For meals, we see the most popular choice for guests is by far the BB option. There are over 90,000 bookings with this option selected, but around 60% of those bookings ended up being canceled. However, the cancelation rates for the other two notable options, HB and SC, seem to have a similar proportion of cancellations as well. This indicates that BB is the highly popular or possibly the most accessible option, and its cancellation rates do not differ vastly from other meal options.

For the rooms data, options A and D are the most popular among guests. While A is the most popular by far, it also has a sizeable proportion of cancellations in comparison to its counterpart D.

In the case of both variables, BB for meals and A for rooms are likely default options that several guests choose without a particularly high preference level. It is likely that cancellation rates are high, not necessarily due to the amenities themselves, but due to the sheer volume of guests picking those options by default when booking a stay.

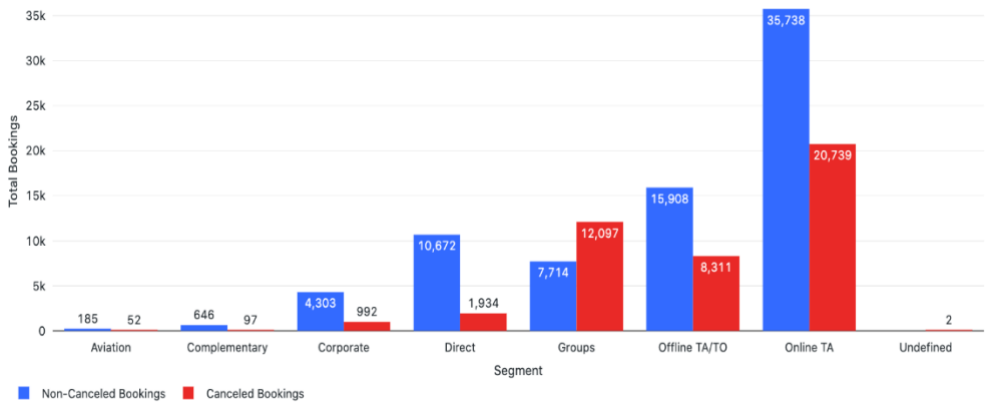
F) How many cancellations come from returning guests?

The number of cancellations from returned guests is 552 compared to 43,672 cancellations from new guests. The cancellation rate for returning guests, 16.94%, is also far lower than that for new guests, 60.73%. This is in part due to the large number of new guests comparatively speaking to the returners, but a lower cancellation rate from guests who have stayed before suggests that they enjoyed the hotel and experienced high-quality service.

The SQL code used to obtain this information is seen below:

```
select is_repeated_guest, is_canceled as Canceled, count(*) as Total
From hotel_bookings
group by is_repeated_guest, Canceled
```

G) Does the market segment a booking belongs to have an impact on whether the stay will be cancelled?



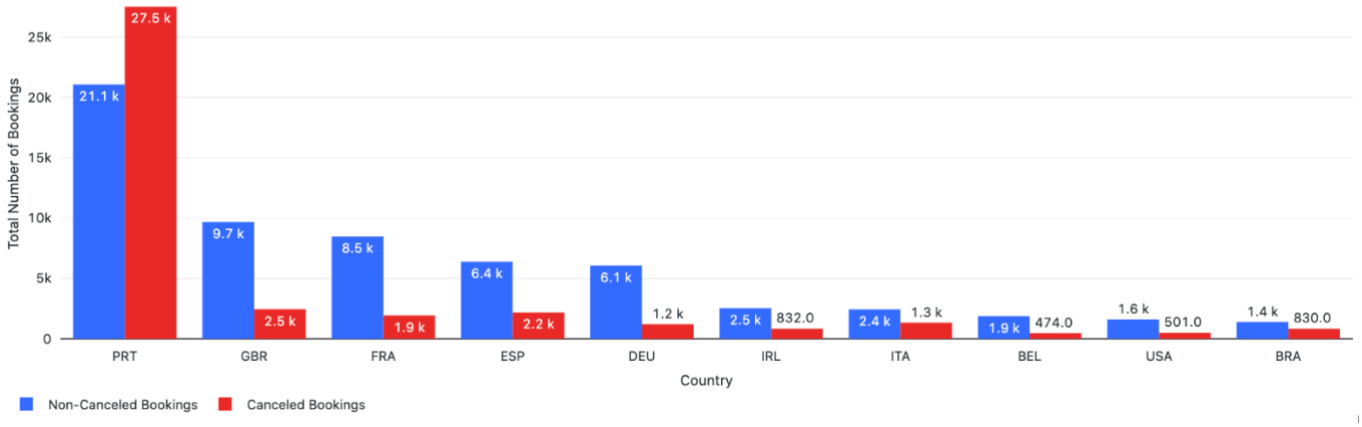
To the left we see the market segments charted by the number of bookings that were canceled and not canceled. The SQL code used to produce this graph is below:

```
select market_segment as
Segment, is_canceled as
Canceled, count(*)as total
from hotel_bookings
group by Segment, Canceled
```

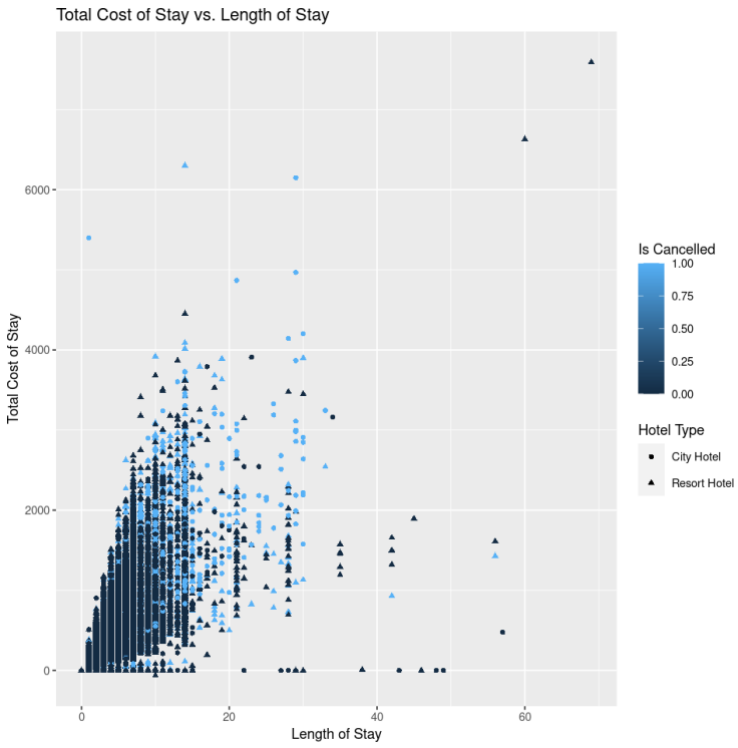
We notice that the Online TA (travel agents), Offline TA/TO (travel agents and tour operators), Groups and Direct segments make up a significant portion of the bookings. Each of these segments has produced more completed bookings than canceled bookings aside from the Groups segment which is in a serious deficit. No data is available about the characteristics of this segment, but some effort should likely be put forth to lower the cancellation rate for that segment, given how many bookings from this group go into the system, but end up producing no revenue.

H) Does the country of origin of the guest have any impact on whether a booking will be canceled?

The visualization below features the top ten countries of origin that had the highest number of cancellations in the data. Both canceled and non-canceled bookings are plotted, and we see that Portugal has the highest number of cancellations and has more canceled bookings than non-canceled bookings. All the other top ten countries have more non-canceled bookings than cancellations with similar proportions, which makes Portugal a special case. *The SQL codes used to generate this information and visualization is found in the appendix under Items M and N.*



I) We expect the length of a stay to be strongly correlated with the cost of the stay, is that accurate?



Calculations were completed to find the total length of each stay and the total amount spent on the stay using the code *found in Item H of the appendix.*

From there a scatter plot was created to visualize the correlation between these values. While we expected a stronger correlation between the two calculated variables, the overall trend is a moderately strong positive correlation. The spread in the data could be attributed to any number of factors, but the points seem to fall in a similar pattern, regardless of whether the hotel was the city or resort Hotel.

We also notice that we see significantly fewer cancellations toward the bottom left where stays are short, and costs are low. We presume that longer, more expensive trips are canceled more often. We also notice some linear intervals of non-canceled stays at the 21-day mark and 28-day mark. These bookings could be work-related trips which often fall upon weeklong intervals and generally try to keep costs low, which is also true for those bookings.

The R code to produce this scatter plot is found in Item I of the appendix.

J) What cost range do the average daily rates fall in for these hotels?

The average value for the variable adr, which indicates the average daily rate for staying at the hotel, was $101.83. The standard deviation was $50.54, indicating that around 68% of our bookings fall within the adr range of $51.29 and $152.37, assuming the prices are normally distributed. The maximum value was well outside that range at $5,400 and the minimum value was –$6.38. We believe this minimum is likely due to a refund given back to a customer for some inconvenience.

The SQL code used to obtain this information if found below:

```
select avg(adr) as Average, min(adr) as Min, max(adr) as Max, stddev(adr) as St_Dev
from hotel_bookings_part;
```

IV. Predicting Cancellations with Classification Modeling

In our analysis of the hotel data, we applied machine learning classification techniques to the data set in order to predict whether a booking is likely to be cancelled or not. The dataset contains various features related to the booking such as arrival and departure dates, lead time, room type, and other customer information.

i. Modeling

We used Gradient Boosting Tree (GBT) algorithm for the classification task. We chose this type of model because GBT is an ensemble learning method that combines weak learners in a sequential manner to improve the model's accuracy. In contrast to other ensemble learning methods that we tried, like Random Forest, GBT builds the model in a stage-wise fashion, where each new weak learner focuses on the samples that were misclassified by the previous weak learners. By utilizing this difference, we knew GBT would significantly reduce the bias and variance of the model and improve the overall predictive power.

Splitting the data into a test and train set was the first step in evaluating the performance of the model. The code to split the data with 70% going into the training set and 30% going into the testing set is found below:

```
%r
#Training Splits
splits <- randomSplit(as.DataFrame(df_final), c(0.7, 0.3), seed = 1123)
train_df <- splits[[1]]
test_df <- splits[[2]]
```

The classification model was trained on the training data, *train_df*, with all variables included and was allowed to be built on a large number of iterations for boosting. Input values for maxDepth and maxIter were manually tuned to find the optimal model:

```
%r
model_gbt <- spark.gbt(train_df,
      is_canceled ~ . ,
      "classification",
      maxDepth = 10,
```

```
maxIter=50,
handleInvalid = "keep")
```

ii. Model Evaluation

We obtained predictions for the test data, test_df, by applying the trained model, and then incorporated them into a new data frame named Output_gbt using the following code. Accuracy was also calculated, which resulted in a rounded value of 1, indicating that nearly 100% of the records were correctly classified.

```
%r
Output_gbt <- predict(model_gbt, test_df)
Correct <- nrow(where(Output_gbt, Output_gbt$is_canceled == Output_gbt$prediction))
Total <- nrow(Output_gbt)
Accuracy = Correct/Total
Accuracy
```

We then used the output of the predictions to evaluate the model's performance using precision, recall, and F1 score for evaluation methods.

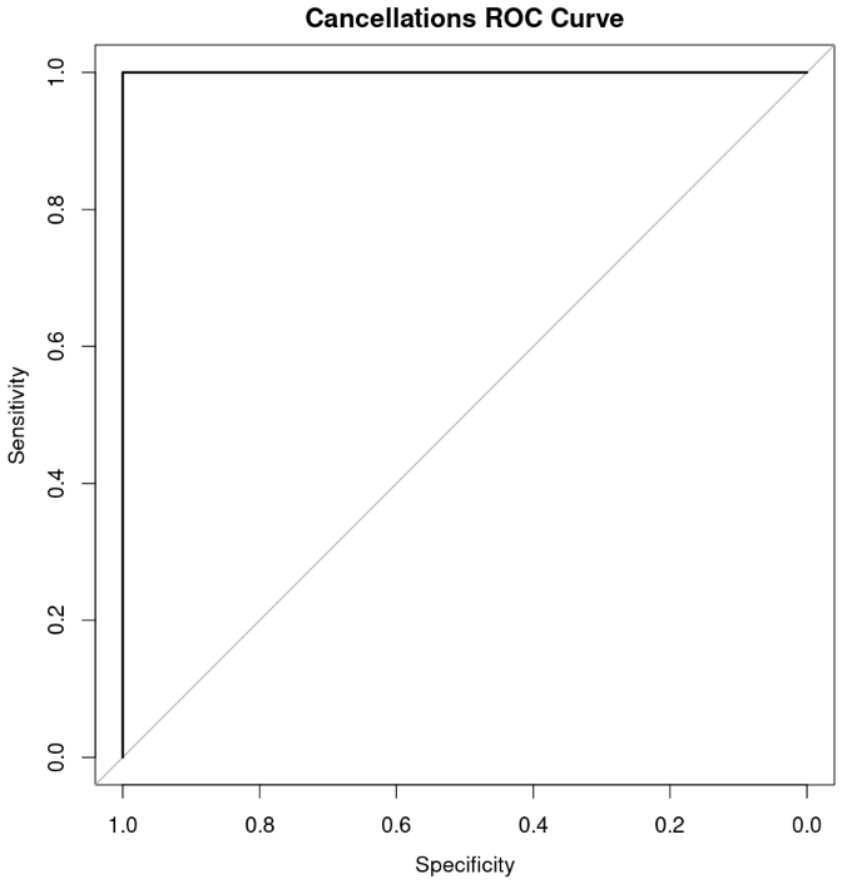
```
%r
TP <- nrow(where(Output_gbt, Output_gbt$is_canceled == 1 & Output_gbt$prediction == 1))
FP <- nrow(where(Output_gbt, Output_gbt$is_canceled == 0 & Output_gbt$prediction == 1))
Precision = TP/(TP+FP)
FN <- nrow(where(Output_gbt, Output_gbt$is_canceled == 1 & Output_gbt$prediction == 0))
Recall = TP/(TP+FN)
F1Score <- 2*((Precision * Recall)/(Precision + Recall))
print(Precision)
print(Recall)
print(F1Score)
```

From our calculations, we were able to obtain values 1, 1, and 1 for precision, recall, and F1 score measures respectively. The real numbers for these values were all rounded up to 1 in the Databricks output and not completely perfect in actuality. Overall, these results demonstrate that our GBT model was effective in predicting hotel booking cancellations and can be a valuable tool for hotels to better manage their reservations and reduce losses from cancellations.

```
%r
library(pROC)
Output_gbt_r=as.data.frame(Output_gbt)
```

```
Output_gbt_r$predPosProb=0 #create a new column

for(i in 1:nrow(Output_gbt_r)){

   Output_gbt_r$predPosProb[i]=Output_gbt_r$probability[[i]][["values"]][[2]]

}

auc(Output_gbt_r$is_canceled, Output_gbt_r$predPosProb)#actual label and predicted prob

rocobj <- plot.roc(Output_gbt_r$is_canceled,Output_gbt_r$predPosProb,main="Cancellations
ROC Curve")
```

We also plotted an ROC curve to further exemplify our model's performance and were able to show our near perfect metrics:



V. Modeling the Average Daily Rate using Regression

To build on our understanding of cancellation rates and their driving factors, our team conducted regression analysis to model the price of a one-night stay in the month of May. This analysis was conducted to identify factors that contribute to cancellations, particularly in the month with a relatively high cancellation rate and the highest reservation volume.

By using this model, we hope to gain insights into how the industry sets pricing and what factors are important for predicting the price of a stay. These insights will help inform the creation of a random forest model aimed at predicting cancellations and aid hotel management in the development of strategies to address the high cancellation rate in May.

iii. Data Preprocessing

The first step in this process was to clean up the dataset to suit the needs of the regression models available for SparkR. Our team did not want any model-specific changes to affect the data used in previous analyses, so a new R data frame to be used specifically for regression was created from the existing R data frame:

```
%r

hotel_regression<- df_final
```

Null values needed to be addressed for the following variables: company, agent, children, and country. The value "NA" replaced any null values that were found in the code below:

```
hotel_regression <- replace(hotel_regression, is.null(hotel_regression), NA)
```

New binary variables were also created for the company and agent columns as well. If the values in these columns were null, a 0 was input into the corresponding new column, and if the values were not null, a 1 was input instead. Additionally, the bookings in the data were reduced to only bookings that occurred in the month of August. The reservation_status_date variable was also dropped as it was unimportant to the analysis. The code for these adjustments is found below:

```
hotel_regression <- hotel_regression%>%

dplyr::mutate(company= ifelse(is.na(company), 0, 1))%>%

dplyr::mutate(agent= ifelse(is.na(agent), 0, 1))%>%

dplyr::filter(adr>1, arrival_date_month == "May")%>%

dplyr::select(-reservation_status_date)

hotel_regression<-na.omit(hotel_regression)
```

Binning the country variable into continent groups was another adjustment that was made. Country codes were combined into lists corresponding to geographic continents and a series of if-else statements were used to create the function applied to the data frame. The new column indicating continent of origin for the hotel guest was appended to the data frame. The code for the binning and function creation is found in the appendix under *Item J* and the code to append the new column is found below:

```
%r

#Update The hotel_df with updated columns

hotel_regression$country <- sapply(hotel_regression$country, country_bin)
```

Some other variables (babies, is_repeated_guest, previous_bookings_not_canceled, and days_in_waiting_list) were deemed unnecessary to the regression analysis by our team and were dropped from the regression dataset using the code below:

```
%r

#Drop Unimportant Variables

hotel_regression<- hotel_regression%>%dplyr::select(-babies, -is_repeated_guest, -
previous_bookings_not_canceled,-days_in_waiting_list)
```

Splitting the data into a test and train set was also necessary to evaluate the performance of the model once the modeling process was complete. The code to split the data with 80% going into the training set and 20% going into the testing set is found below:

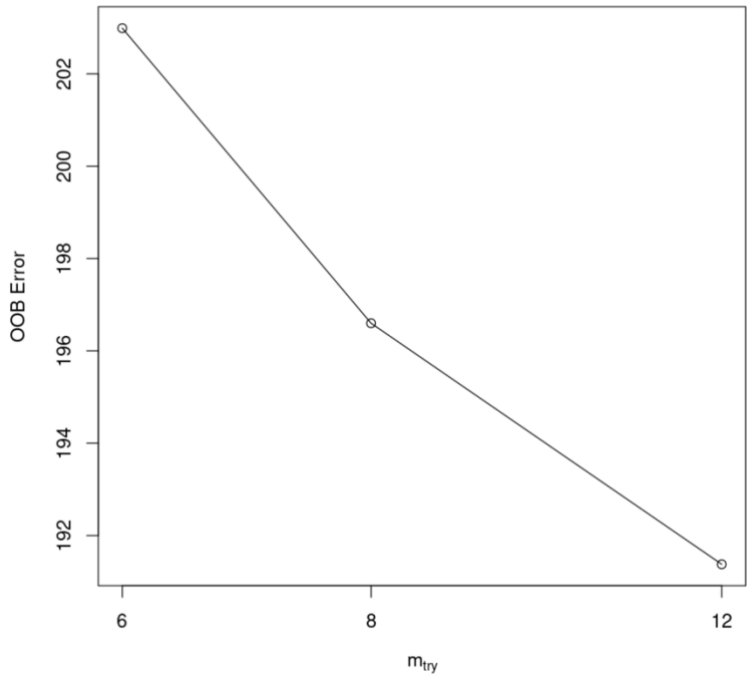
```
%r

#Training Splits

splits <- randomSplit(hotel_regression, c(0.8, 0.2), seed = 123)

train_df <- splits[[1]]

test_df <- splits[[2]]
```

iv. Modeling

Our team decided to implement the Random Forest Regression model available in R to model the price of a one-night stay in August. Instead of explicitly selecting the number of variables for use in building a decision tree, we instead chose to let the model sample numbers for mtry to reduce MSE. Random Forests' tuneRF function was utilized to find the optimal value for mtry to input into the model. mtry is the "number of variables randomly sampled as candidates at each split" according to R's documentation for randomForest. This input for the model was optimized using the code below and the optimal value for mtry is highlighted in the graph below:

```
%r
# Finding The Optimal Max mtry
tuneRF(
x = train_df%>%dplyr::select(-adr),
y = train_df$adr,
ntreeTry = 500,
stepFactor = 1.5,
improve = 0.1,
plot= TRUE
)
```

The function determined that the value of 13 for mtry minimized the overall error which can be seen in the graph.



The regression model was trained on the training data, *train_df*, with all variables included and the optimal values for number of trees, depth of tree, and mtry. Optimal values for number of trees and depth of tree were manually tuned to find the inputted values in the code below:

```
%r

#ntree plateaus at 500, maxdepth: 10, mtry at 12

#Regression Model Random Forest

model_rfr <- randomForest(adr ~ .,

data = train_df,

ntree = 500,

maxdepth = 10,
```

```
mtry=12)
```

v. Model Evaluation

Predictions for the testing data, *test_df*, using the trained model were obtained and inserted into a new data frame *Output_rfr* using the code below:

```
%r

predictions<-predict(model_rfr , test_df)

Output_rfr <- cbind(test_df, predictions)

Output_rfr<- createDataFrame(Output_rfr)
```

Several performance measures were then calculated using the output from the model's application on the testing data. The code to obtain the values for MSE, RMSE, MAE, and MAPE is found below:

```
%r

#Calculate MSE, RMSE, MAE and MAPE on the testing set

mse<- showDF(SparkR::select(Output_rfr, avg((Output_rfr$adr -
Output_rfr$predictions)^2)))

rmse<- showDF(SparkR::select(Output_rfr, sqrt(avg((Output_rfr$adr-
Output_rfr$predictions)^2))))

mae<- showDF(SparkR::select(Output_rfr, mean(abs(Output_rfr$adr -
Output_rfr$predictions))))

mape<- showDF(SparkR::select(Output_rfr, mean(abs((Output_rfr$adr -
Output_rfr$predictions) / Output_rfr$adr))))
```

The MSE value for the model was roughly 524.84 which is a relatively high amount of error, but the RMSE value for the model was roughly 22.9094. The average value for adr, the target variable, is $108.70 in the month of May. Given that average price, a $22.91 margin of error in predicting price is not necessarily ideal, but is not completely off base either. The MAE value for the model was roughly 16.5822. Our MAE is significantly lower than our RMSE, which indicates to us that we have a few bookings where the predicted values are very different from the actual values. These bookings that produced large errors are weighted more heavily in the RMSE but are weighted evenly with all other bookings in the MAE. From this we can conclude that our model is overall doing an acceptable job with a margin of error of around $16.58, but there are a few bookings with large errors that make that margin of error less reliable in some cases. The MAPE value for the model was roughly 0.1617 meaning that our predicted values are on average about 16% off from the actual value. This once again is not an ideal value for MAPE, but a reasonable margin of error given we have some perceived outliers that the model does not do a great job predicting.

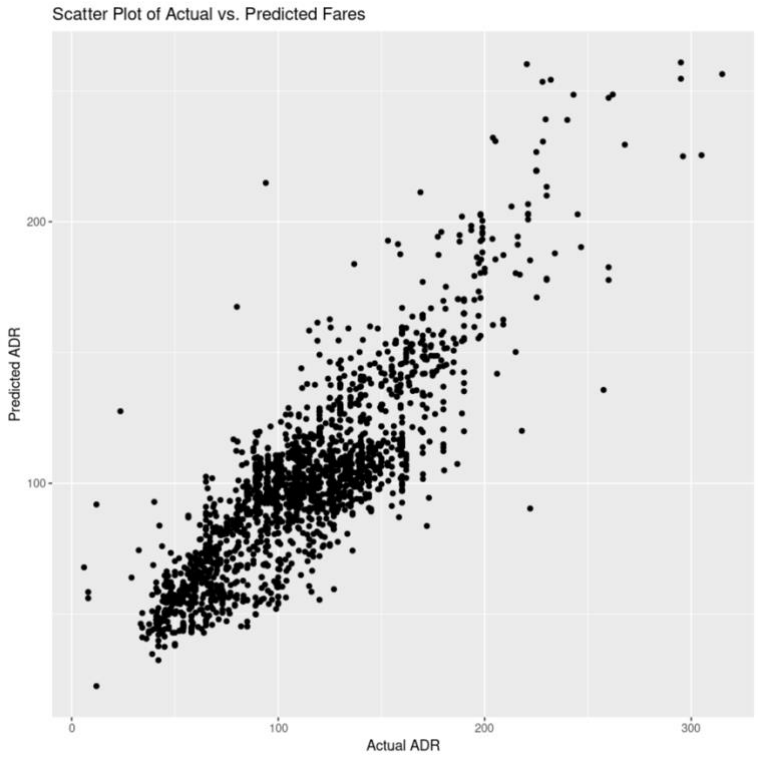
The Pseudo-R squared value for the model was also obtained and the code for that portion is found below:

```
%r

#Pseudo - R^2

Output_rfr<-as.data.frame(Output_rfr)

model <- lm(Output_rfr$adr ~ Output_rfr$predictions)

# Print the R-squared value

summary(model)$r.squared
```

The R squared value for the model was roughly 0.7458, indicating that the model explained 74.58% of the variation in the average daily rate using the variables included in the model. Overall, that is a relatively high R squared value and we can be confident that we have a fairly robust model to predict the price of a one-night stay in May.

Given that we have a decent margin of error based on our performance measures, we visualized the errors by plotting the actual values of the test data against the predicted values.

Overall, most of the data points are predicted well with values positively correlated. There are some outlying data points on both the low and high end of the average daily rate scale, with some visibly large errors for higher ADRs. The large errors falling on the high end of the scale indicates that this model struggles to predict the value of high ADRs than it does for low ones. It is important to note that this model is more effective at predicting lower costs.



The code to obtain this scatter plot is found below:

```
%r
ggplot(as.data.frame(Output_rfr),
aes(x = adr, y = predictions)) +
geom_point() +
labs(x = "Actual ADR",
y = "Predicted ADR",
title = "Scatter Plot of Actual vs.
Predicted Fares")
```

VI. Conclusion

The Gradient Boosting Tree classification model and the Random Forest regression model are both useful tools for hotel management to utilize. For cancellation classification, accuracy, precision, recall, and the F1 score were all very high and rounded to 1 in the Databricks notebook. A model was developed that will allow management to accurately predict the likelihood of a hotel reservation cancellation and adjust occupancy rates based on the different parameters indicated. This will lead to more revenue for the hotel. However, there may be some form of skewness which might have led to some form of overfitting and a form of bias, hence the perfect accuracy. The nature of the data set may have caused this skewness. The dataset had some exceptions and may have been incomplete. Also, the algorithm may have singled out some factors as more likely for a cancellation than other factors. The regression analysis on the average daily rate for May resulted in a fairly robust model that does well at predicting low average daily rates and struggles at predicting high average daily rates. The model explains around 74.58% of the variation in ADR for the strongest month for bookings by using all of the variables provided in the dataset. Both of these tools can be utilized to increase revenue and profits.

VII. Appendix

Item A: A list of variables included in the dataset and their descriptions:

Variable:	Description:
hotel	Whether the hotel is the city hotel or the resort hotel
is_canceled	Binary variable indicating if the booking was canceled or not
lead_time	Number of days between the day the stay was booked and the arrival date
arrival_date_year	Year of arrival date
arrival_date_month	Month of arrival date
arrival_date_week_number	Week number of year for arrival date
arrival_date_day_of_month	Day of month for arrival date
stays_in_weekend_nights	Number of weekend nights (Saturday and Sunday) guest stayed or booked to stay at the hotel
stays_in_week_nights	Number of week nights (Monday – Friday) the guest stayed or booked to stay at the hotel
adults	Number of adults for booking
children	Number of children for booking
babies	Number of babies for booking
meal	Type of meal booked
country	Guest's country of origin
market_segment	Market segment designation
distribution_channel	Distribution channel in which the stay was booked
is_repeated_guest	Binary variable indicating if the guest had previously stayed at the hotel or not
previous_cancellations	Number of bookings that were canceled by the guest prior to this booking record
previous_bookings_not_canceled	Number of bookings that were not canceled by the guest prior to this booking record
reserved_room_type	Type of room reserved by the guest
assigned_room_type	Type of room the guest received
booking_changes	Number of adjustments that were made to the booking after the initial reservation was made
deposit_type	Indicates whether a customer made a deposit to guarantee the booking
agent	The ID for the travel agency that made the booking
company	The ID for the company that made the booking
days_in_waiting_list	Number of days the booking was in the waiting list before it was confirmed for the guest
customer_type	Type of booking
adr	Average Daily Rate: the sum of all transactions made by the guest at the hotel divided by the number of staying nights
required_car_parking_spaces	Number of car parking spaces required by the guest
total_of_special_requests	Number of special requests made by the guest
reservation_status	The last recorded status of the reservation

reservation_status_data	The date when the last recorded status of the reservation was set

Item B: Hive SQL code to create the Hive table containing the data from the CSV file, hotel_bookings

```
use vikyadav;
DROP TABLE IF EXISTS hotel_bookings;
CREATE TABLE IF NOT EXISTS hotel_bookings (
ID INT,
hotel VARCHAR(50),
is_canceled INT,
lead_time INT,
arrival_date_year INT,
arrival_date_month VARCHAR(20),
arrival_date_week_number INT,
arrival_date_day_of_month INT,
stays_in_weekend_nights INT,
stays_in_week_nights INT,
adults INT,
children INT,
babies INT,
meal VARCHAR(20),
country VARCHAR(50),
market_segment VARCHAR(50),
distribution_channel VARCHAR(50),
is_repeated_guest INT,
previous_cancellations INT,
previous_bookings_not_canceled INT,
reserved_room_type VARCHAR(10),
assigned_room_type VARCHAR(10),
booking_changes INT,
deposit_type VARCHAR(20),
agent INT,
company INT,
days_in_waiting_list INT,
```

```
customer_type VARCHAR(20),
adr DOUBLE,
required_car_parking_spaces INT,
total_of_special_requests INT,
reservation_status VARCHAR(20),
reservation_status_date STRING)
USING CSV
LOCATION "/users/vikyadav/hotel_bookings.csv"
OPTIONS("header" = "true");
```

Item C: SQL code to create the new partitioned table, hotel_bookings_part, using the existing hotel_bookings table

```
DROP TABLE IF EXISTS hotel_bookings_part;
CREATE TABLE IF NOT EXISTS hotel_bookings_part (
ID INT,
hotel VARCHAR(50),
is_canceled INT,
lead_time INT,
arrival_date_year INT,
arrival_date_month VARCHAR(20),
arrival_date_week_number INT,
arrival_date_day_of_month INT,
stays_in_weekend_nights INT,
stays_in_week_nights INT,
adults INT,
children INT,
babies INT,
meal VARCHAR(20),
country VARCHAR(50),
market_segment VARCHAR(50),
distribution_channel VARCHAR(50),
is_repeated_guest INT,
previous_cancellations INT,
previous_bookings_not_canceled INT,
```

```
reserved_room_type VARCHAR(10),
assigned_room_type VARCHAR(10),
booking_changes INT,
deposit_type VARCHAR(20),
agent INT,
company INT,
days_in_waiting_list INT,
customer_type VARCHAR(20),
adr DOUBLE,
required_car_parking_spaces INT,
total_of_special_requests INT,
reservation_status VARCHAR(20),
reservation_status_date string)
PARTITIONED BY (res_year INT, res_month VARCHAR(20));
```

Item D: R code for analysis of the hotel variable and the is_canceled variable

```
%r
ggplot(data = df_final,
aes(
x = hotel,
y = prop.table(stat(count)),
fill = factor(is_canceled),
label = scales::percent(prop.table(stat(count)))
)) +
geom_bar(position = position_dodge()) +
geom_text(
stat = "count",
position = position_dodge(.9),
vjust = -0.5,
size = 3
) +
scale_y_continuous(labels = scales::percent) +
labs(title = "Reservation Status by Hotel Type", x = "Hotel Type", y = "Count") +
```

```
theme_classic() +
scale_fill_discrete(
name = "Reservation Status",
breaks = c("0", "1"),
labels = c("Not Cancelled", "Cancelled")
)
```

Item E: SQL code for analysis of the lead_time variable

```
SELECT LeadtimeRange, Total_Bookings, Canceled_Bookings, Total_Bookings-
Canceled_Bookings as uncanceled_bookings, Canceled_Bookings/Total_Bookings*100 as
Percent_Canceled
FROM(
SELECT
case
when lead_time <= 7 THEN "Within the Week"
when lead_time>7 AND lead_time<30 THEN "Within the Month"
when lead_time>30 AND lead_time<=180 THEN "Within 6 Months"
when lead_time>180 AND lead_time<360 THEN "Within the Year" ELSE "Over a Year" END
LeadtimeRange,
count(*) as Total_Bookings, sum(is_canceled) as Canceled_Bookings
FROM hotel_bookings_part
GROUP BY 1
ORDER BY Case WHEN LeadTimeRange = "Within the Week" THEN 1
WHEN LeadTimeRange = "Within the Month" THEN 2
WHEN LeadTimeRange = "Within 6 Months" THEN 3
WHEN LeadTimeRange = "Within the Year" THEN 5
WHEN LeadTimeRange = "Over a Year" THEN 6 END asc);
```

Item F: SQL code for analysis of the adults, children, and babies variables

```
SELECT Demographic, Total_Bookings, Canceled_Bookings, Total_Bookings-Canceled_Bookings
as Noncanceled_Bookings, Canceled_Bookings/Total_Bookings*100 as Percent_Canceled
FROM(
select count(*) as Total_Bookings, sum(is_canceled) as Canceled_Bookings,
CASE WHEN children >= 1 THEN "Family"
```

```
WHEN children < 1 AND adults = 1 THEN "Single"
WHEN children < 1 AND adults = 2 THEN "Couple"
WHEN children < 1 AND adults > 2 THEN "Group"
WHEN babies >= 1 THEN "Family"
WHEN babies < 1 AND adults = 1 THEN "Single"
WHEN babies < 1 AND adults = 2 THEN "Couple"
WHEN babies < 1 AND adults > 2 THEN "Group"
ELSE "Other" END as Demographic
FROM hotel_bookings_part
GROUP BY Demographic)
```

Item G: R code used to create a stacked bar chart analysis of hotel room types and cancellations

```
%r
room <-df_final %>%
group_by(is_canceled, assigned_room_type) %>%
summarise(bookings_count = n()) %>%
ungroup()
df_room <- collect(room)
df_room$is_canceled<-ifelse(df_room$is_canceled=="1","Yes","No")
ggplot(df_room, aes(x=assigned_room_type, y=bookings_count, fill=is_canceled)) +
geom_bar(position='stack', stat='identity') +
scale_fill_manual(values=c('blue', 'red')) +
labs(title = "Total Bookings for Each Room Type Colored by Cancellations", x = "Room
Type", y = "Bookings Count", fill = "Is Canceled?")
```

Item H: R code to find total length of stay and total cost of stay

```
%r
hotel_data <- df_final %>%
mutate(adr = replace(adr, adr>500, mean(adr)))%>%
mutate(stay_nights_total = stays_in_weekend_nights + stays_in_week_nights,
stay_cost_total = adr * stay_nights_total)
summary(hotel_data$stay_nights_total)
summary(hotel_data$stay_cost_total)
```

Item I: R code to plot length of stay and cost of stay data on a scatter plot

```
%r

scatter <- ggplot(hotel_data,

aes(x=stay_nights_total,

y=stay_cost_total,

shape=hotel,

color=is_canceled))+ geom_point(alpha=1)

scatter + labs(x = "Length of Stay", y = "Total Cost of Stay", title = "Total Cost of
Stay vs. Length of Stay", color = "Is Cancelled", shape = "Hotel Type")
```

Item J: R code binning the country values into continent groups and creating the corresponding function to apply to regression dataset

```
%r

#Binning The Countries Into Continents

europe=c('PRT','GBR','ESP','IRL','FRA','ROU','NOR','POL','DEU','BEL','CHE','GRC','ITA','
NLD','DNK','RUS','SWE','EST','CZE','FIN','LUX','SVN','ALB','UKR','SMR','LVA','SRB','AUT'
,'BLR','LTU','TUR','HUN','HRV','GEO','AND','SVK','MKD','BIH','BGR','MLT','ISL','MCO','LI
E','MNE')

north_a=c('USA','MEX','PRI','CRI','CUB','HND','NIC','GAB','PAN','SLV','GTM')

south_a=c('ARG','BRA','CHL','URY','COL','VEN','SUR','PER','ECU','BOL','PRY','GUY')

asia=c('OMN','CN','IND','CHN','ISR','KOR','ARE','HKG','IRN','CYP','KWT','MDV','KAZ','PAK
','IDN','LBN','PHL','AZE','BHR','THA','MYS','ARM','JPN','LKA','JOR','SYR','SGP','SAU','V
NM','QAT','UZB','NPL','MAC','TWN','IRQ','KHM','BGD','TJK','TMP','MMR','LAO')

africa=c('MOZ','BWA','MAR','ZAF','AGO','ZMB','ZWE','DZA','TUN','CAF','NGA','SEN','SYC','
CMR','MUS','COM','UGA','CIV','BDI','EGY','MWI','MDG','TGO','DJI','STP','ETH','RWA','BEN'
,'TZA','GHA','KEN','GNB','BFA','LBY','MLI','NAM','MRT','SDN','SLE')

australia=c('AUS')

Others=c('CYM','CPV','JAM','GIB','JEY','GGY','FJI','NZL','DOM','PLW','BHS','KNA','IMN','
VGB','GLP','UMI','MYT','FRO','BRB','ABW','AIA','DMA','PYF','LCA','ATA','ASM','NCL','KIR'
,'ATF')

unk=c('Unknown')

country_bin <- function(x) {

if (x %in% europe) {

return('Europe')

} else if (x %in% north_a) {

return('North America')
```

```
} else if (x %in% south_a) {
return('South America')
} else if (x %in% asia) {
return('Asia')
} else if (x %in% africa) {
return('Africa')
} else if (x %in% australia) {
return('Australia')
} else if (x %in% Others) {
return('Others')
} else if (x %in% unk) {
return('Unknown')
}
}
```

Item K: SQL code used to obtain total bookings for each month in each year of the data

```
select arrival_date_month, arrival_date_year, is_canceled, count(*) as Total,
CASE WHEN arrival_date_month = 'January' THEN 1
WHEN arrival_date_month = 'February' THEN 2
WHEN arrival_date_month = 'March' THEN 3
WHEN arrival_date_month = 'April' THEN 4
WHEN arrival_date_month = 'May' THEN 5
WHEN arrival_date_month = 'June' THEN 6
WHEN arrival_date_month = 'July' THEN 7
WHEN arrival_date_month = 'August' THEN 8
WHEN arrival_date_month = 'September' THEN 9
WHEN arrival_date_month = 'October' THEN 10
WHEN arrival_date_month = 'November' THEN 11
WHEN arrival_date_month = 'December' THEN 12 END as Month_F
FROM hotel_bookings
GROUP BY arrival_date_month, arrival_date_year, is_canceled
ORDER BY Month_F asc;
```

Item L: SQL code used to obtain average bookings and average cancellations for each month in the data

```
SELECT res_month, Average_Bookings, Average_Canceled,
Average_Canceled/Average_Bookings*100 as Percent_Canceled, Average_Bookings-
Average_Canceled as Average_Noncanceled,

CASE WHEN res_month = 'January' THEN 1

WHEN res_month = 'February' THEN 2

WHEN res_month = 'March' THEN 3

WHEN res_month = 'April' THEN 4

WHEN res_month = 'May' THEN 5

WHEN res_month = 'June' THEN 6

WHEN res_month = 'July' THEN 7

WHEN res_month = 'August' THEN 8

WHEN res_month = 'September' THEN 9

WHEN res_month = 'October' THEN 10

WHEN res_month = 'November' THEN 11

WHEN res_month = 'December' THEN 12 END as Month_F

FROM

(SELECT res_month, avg(Total_Bookings) as Average_Bookings, avg(Canceled_Bookings) as
Average_Canceled

FROM(

SELECT res_month, res_year, count(*) as Total_Bookings, sum(is_canceled) as
Canceled_Bookings

FROM hotel_bookings_part

GROUP BY res_month, res_year)

GROUP BY res_month)

ORDER BY Month_F;
```

Item M: SQL code used to obtain the countries of origin with the largest amounts of cancellations

```
select country, is_canceled as Canceled, count(*) as total

from hotel_bookings

Where is_canceled = 1

group by country, Canceled

order by total desc

limit 10

-- countries with highest number of cancellations
```

Item N: SQL code used to create bookings visualization for the top ten countries of origin with the most cancellations

```
select country, is_canceled as Canceled, count(*) as total

from hotel_bookings

where country == 'PRT' OR country == 'GBR' OR country == 'ESP' OR country == 'FRA' OR
country == 'ITA' OR country == 'DEU' OR country == 'IRL' OR country == 'BRA' OR country
== 'USA' OR country == 'BEL'

group by country, Canceled

order by total desc
```

Dataset link: https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand?resource=download

VIII. Contributions

Ifeoma: Worked on writing up the final report and worked on data summarization visualizations.

Matteo: Completed the classification analysis and worked on writing up the final report.

Evan: Completed the regression analysis and loaded data into Hive tables.

Somer: Worked on writing up the final report and worked on data summarization visualizations.

Vikas: Completed data summarization, made visualizations, and loaded data into Hive tables.