Predicting Hotel Cancellations

Brendan Cheng, Alex Szeto, Chlinton Kuang, Dylan Sevilla

2023-06-09

- Contributions
- Introduction
- Loading Data and Libraries
- Data Cleaning and Preprocessing
- EDA
    - 1. What is the busiest months/seasons for hotel bookings? Are the busiest months/seasons correlated with more expensive prices?
    - 2. What percentage of stays are booked in advance? How far in advance are they booked and is that correlated with the likelihood of cancellation?
    - 3. What continents/region are people booking from? Which season do the most people travel from each continent/region?
    - 4. How does the type of room correlate with cancellation?
    - 5. Does the number of guests affect the likelihood of a booking being cancelled?
    - 6. Do cancellations occur in greater proportion the more days are booked?
    - 7. Are business trips more likely to have hotel cancellations?
    - 8. Does the days booked in advance affect hotel cancellation?
- Predictive Models
    - Logistic Regression
    - Decision Tree
    - KNN
- Findings and Conclusions

Contributions

- Brendan Cheng
  - EDA (3, 4)
  - Logistic Regression
  - KNN
- Alex Szeto
  - EDA (7, 8)
  - Decision Tree
- Chlinton Kuang
  - EDA (5, 6)
  - KNN
- Dylan Sevilla
  - EDA (1, 2)

Introduction

Our data is from 2 hotels in Portugal, the City Hotel and the Resort Hotel. We have 119390 bookings from 2015-2017 with 32 variables. The variables contain a variety of information about each booking made, including the number of guests, the nightly rate, the date of the room reservation, etc. We would like to determine which of the variables are relevant in predicting whether a booking will be cancelled, and build predictive models using such variables.

The datasets used can be accessed through the following links:

- hotel booking data (https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand)
- regional codes to classify countries into regions/continents (https://github.com/lukes/ISO-3166-Countries-with-Regional-Codes/blob/master/all/all.csv)

Loading Data and Libraries

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.3
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
## Warning: package 'tibble' was built under R version 4.1.3
```

```
## Warning: package 'tidyr' was built under R version 4.1.3
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```
## Warning: package 'purrr' was built under R version 4.1.3
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
## Warning: package 'stringr' was built under R version 4.1.3
```

```
## Warning: package 'forcats' was built under R version 4.1.3
```

```
## Warning: package 'lubridate' was built under R version 4.1.3
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.3
```

```
## Warning: package 'lattice' was built under R version 4.1.3
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.1.3
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.1.3
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.1.3
```

```
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.1.3
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.1.3
```

```
hotel_bookings = read_csv("hotel_bookings.csv", show_col_types = FALSE)
country_codes = read_csv("country_codes.csv", show_col_types = FALSE)
```

Data Cleaning and Preprocessing

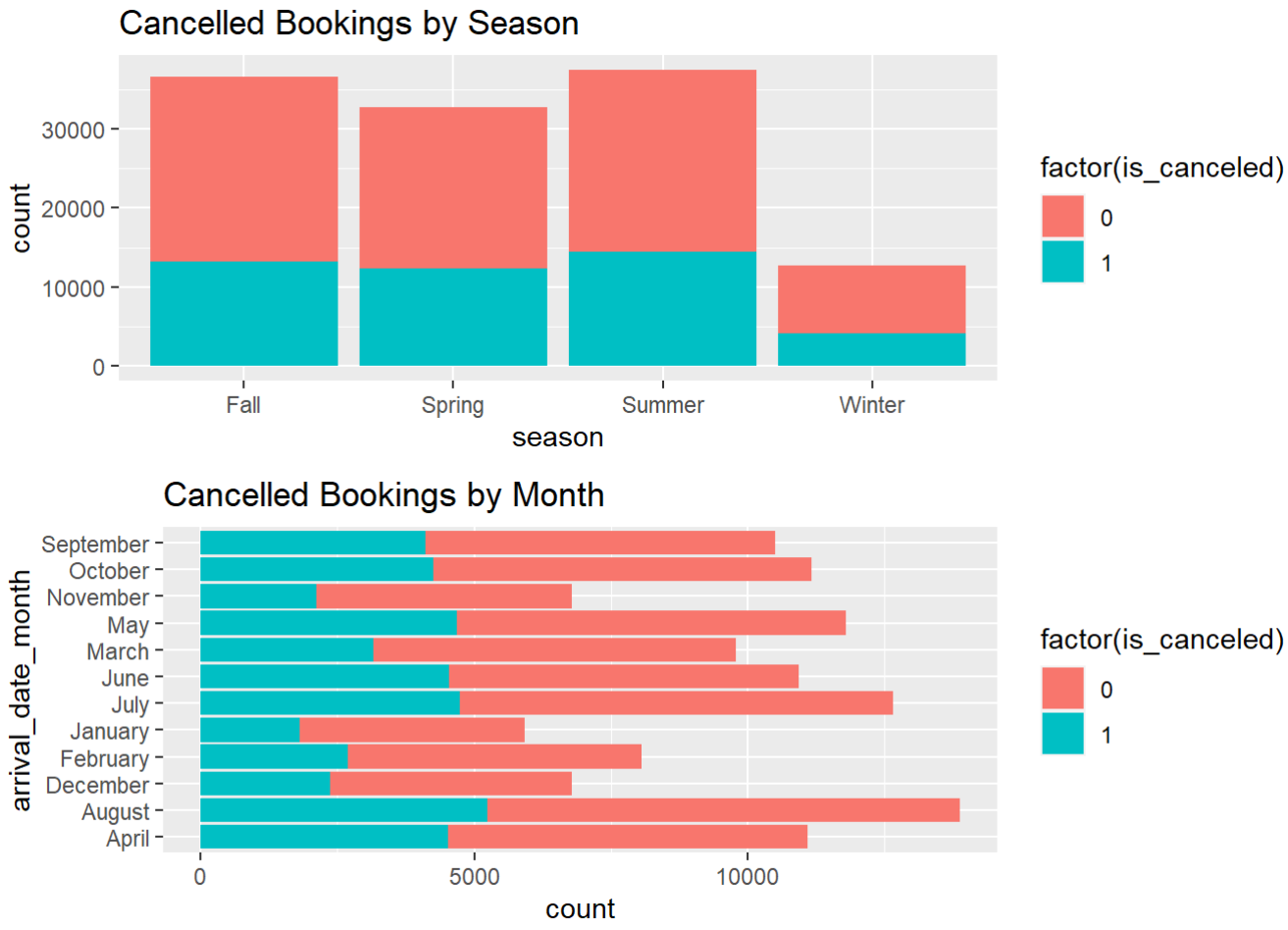
```
hotel_bookings = hotel_bookings %>%
    mutate(country = ifelse(country == "CN", "CHN", country))

country_codes = country_codes %>% select("name", "alpha-3", "region", "sub-region") %>%
    rename("country name" = "name", "subregion" = "sub-region")

# attach country data to bookings data
hotel_bookings = left_join(hotel_bookings, country_codes, by = c("country" = "alpha-3"))

hotel_bookings = hotel_bookings %>%
    mutate(season = case_when(arrival_date_month %in%
                                  c("December", "January", "Feburuary") ~ "Winter",
                              arrival_date_month %in%
                                  c("March", "April", "May") ~ "Spring",
                              arrival_date_month %in%
                                  c("June","July", "August") ~ "Summer",
                              .default = "Fall"))
hotel_bookings = hotel_bookings %>%
  mutate(num_nights = stays_in_weekend_nights + stays_in_week_nights, total_cost = adr*num_ni
ghts)
dim(hotel_bookings)
```

```
## [1] 119390     38
```

EDA

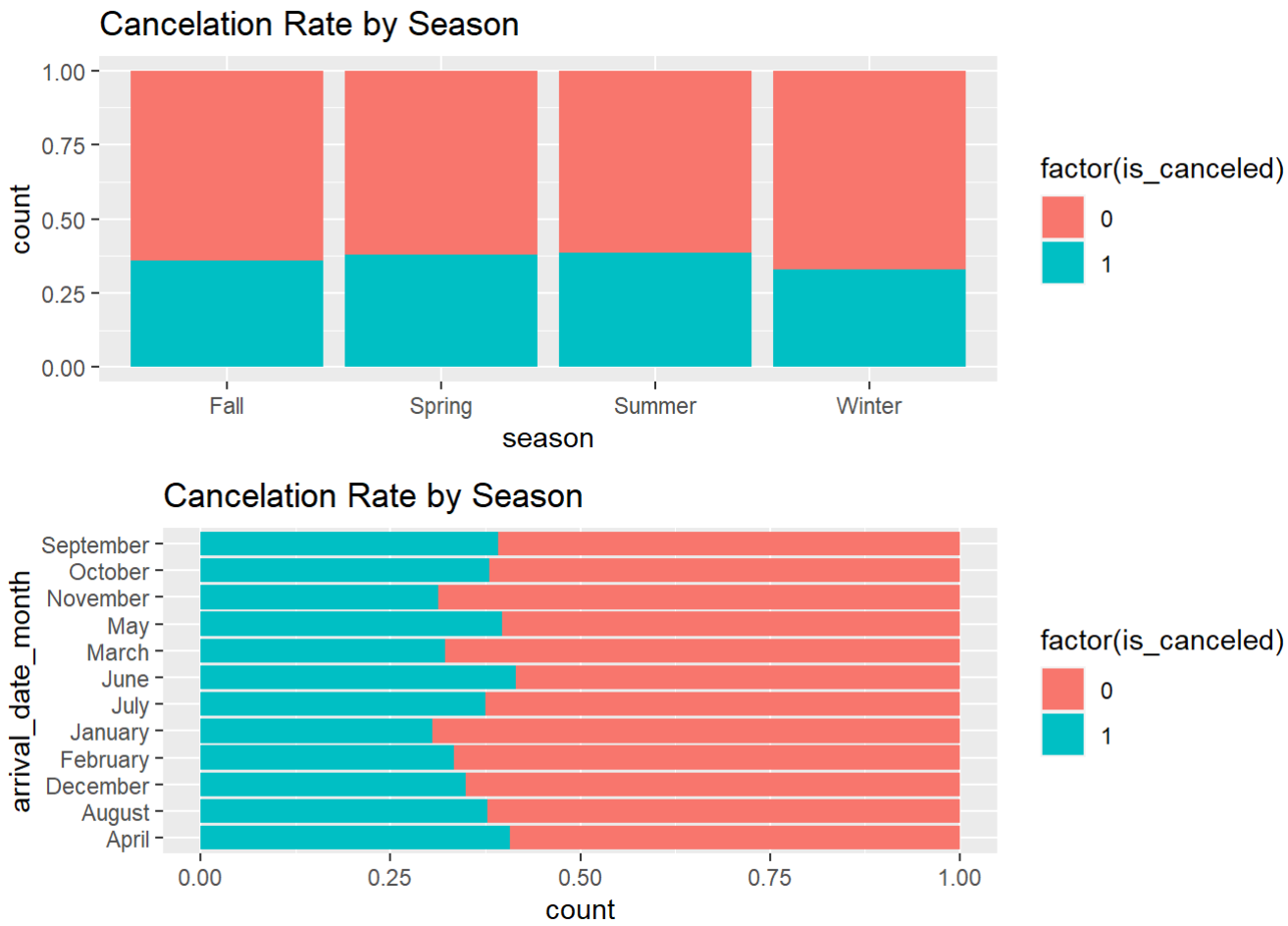
1. What is the busiest months/seasons for hotel bookings? Are the busiest months/seasons correlated with more expensive prices?

```
fig1 = ggplot(hotel_bookings) +
    geom_bar(aes(x = season, group = is_canceled, fill = factor(is_canceled))) +
    labs(title = "Cancelled Bookings by Season")
fig2 = ggplot(hotel_bookings) +
    geom_bar(aes(x = arrival_date_month, group = is_canceled, fill = factor(is_canceled))) +
    labs(title = "Cancelled Bookings by Month") + coord_flip()

grid.arrange(fig1, fig2)
```



Summer is the busiest season with most of the days being reserved for August. Winter is off-peak season with barely any bookings compared to the other 3 seasons.

```
fig3 = ggplot(hotel_bookings) +
    geom_bar(aes(x = season, group = is_canceled, fill = factor(is_canceled)),
             position = "fill") +
    labs(title="Cancelation Rate by Season")
fig4 = ggplot(hotel_bookings) +
    geom_bar(aes(x = arrival_date_month, group = is_canceled, fill = factor(is_canceled)),
             position = "fill") +
    labs(title = "Cancelation Rate by Season")+ coord_flip()

grid.arrange(fig3, fig4)
```

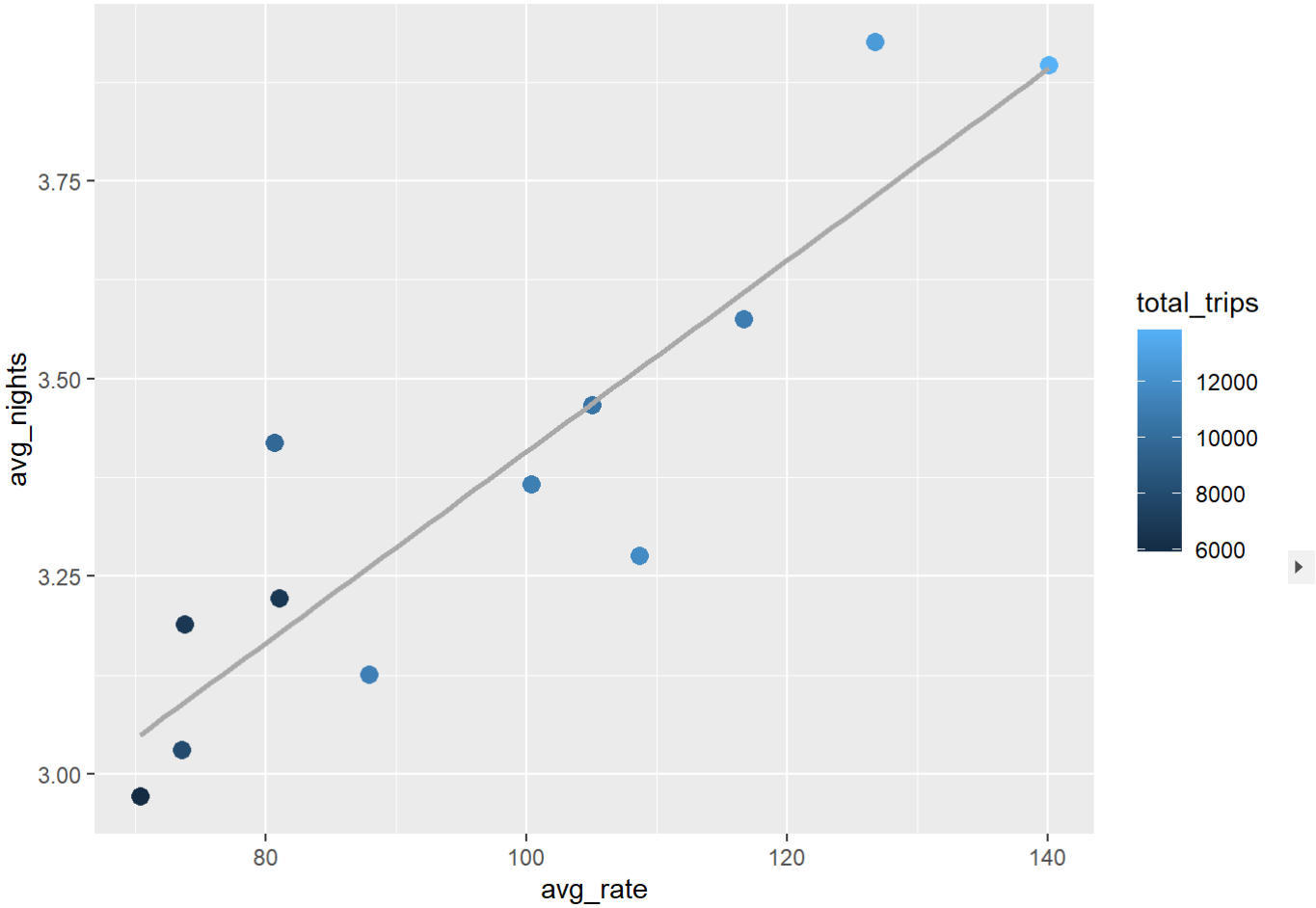


The cancellation rate is very consistent throughout each season and/or month.

arrival_date_month <chr>	total_canceled <dbl>	total_trips <int>	cancel_rate <dbl>	avg_rate <dbl>	avg_cost <dbl>	avg_
April	4524	11089	0.4079719	100.38079	334.6078	3.
August	5239	13877	0.3775312	140.11152	572.5097	3.
December	2371	6780	0.3497050	81.07678	265.5151	3.
February	2696	8068	0.3341596	73.58228	225.4552	3
January	1807	5929	0.3047731	70.36124	214.7204	2.
July	4742	12661	0.3745360	126.78801	520.3154	3.
June	4535	10939	0.4145717	116.67219	411.3161	3.
March	3149	9794	0.3215234	80.67965	269.4733	3.

arrival_date_month <chr>	total_canceled <dbl>	total_trips <int>	cancel_rate <dbl>	avg_rate <dbl>	avg_cost <dbl>	avg_
May	4677	11791	0.3966585	108.69552	348.4120	3.
November	2122	6794	0.3123344	73.79496	235.5250	3.

1-10 of 12 rows Previous **1** 2 Next



The graph shows that the months with higher amounts of bookings also tend to have higher nightly rates as well as longer trips. In tandem, both of these trends indicate that trips during busier months tend to be more expensive.

2. What percentage of stays are booked in advance? How far in advance are they booked and is that correlated with the likelihood of cancellation?

```
bookings_advance = hotel_bookings %>% mutate(book_adv = lead_time != 0) %>%
  select(is_canceled, lead_time, book_adv)

mean(bookings_advance$book_adv)
## [1] 0.9468548
mean(bookings_advance$lead_time)
## [1] 104.0114
median(bookings_advance$lead_time)
## [1] 69
```

```
##
## Call:
## glm(formula = is_canceled ~ lead_time, family = binomial, data = bookings_advance)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5265  -0.8797  -0.7481   1.2302   1.6952
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.166e+00  9.182e-03  -126.9   <2e-16 ***
## lead_time    5.855e-03  6.137e-05    95.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 157398  on 119389  degrees of freedom
## Residual deviance: 147158  on 119388  degrees of freedom
## AIC: 147162
##
## Number of Fisher Scoring iterations: 4
```

```
##
##         FALSE  TRUE
##   FALSE 66662  8504
##   TRUE  31938 12286
```

```
## [1] "Accuracy: 0.661261412178574"
```

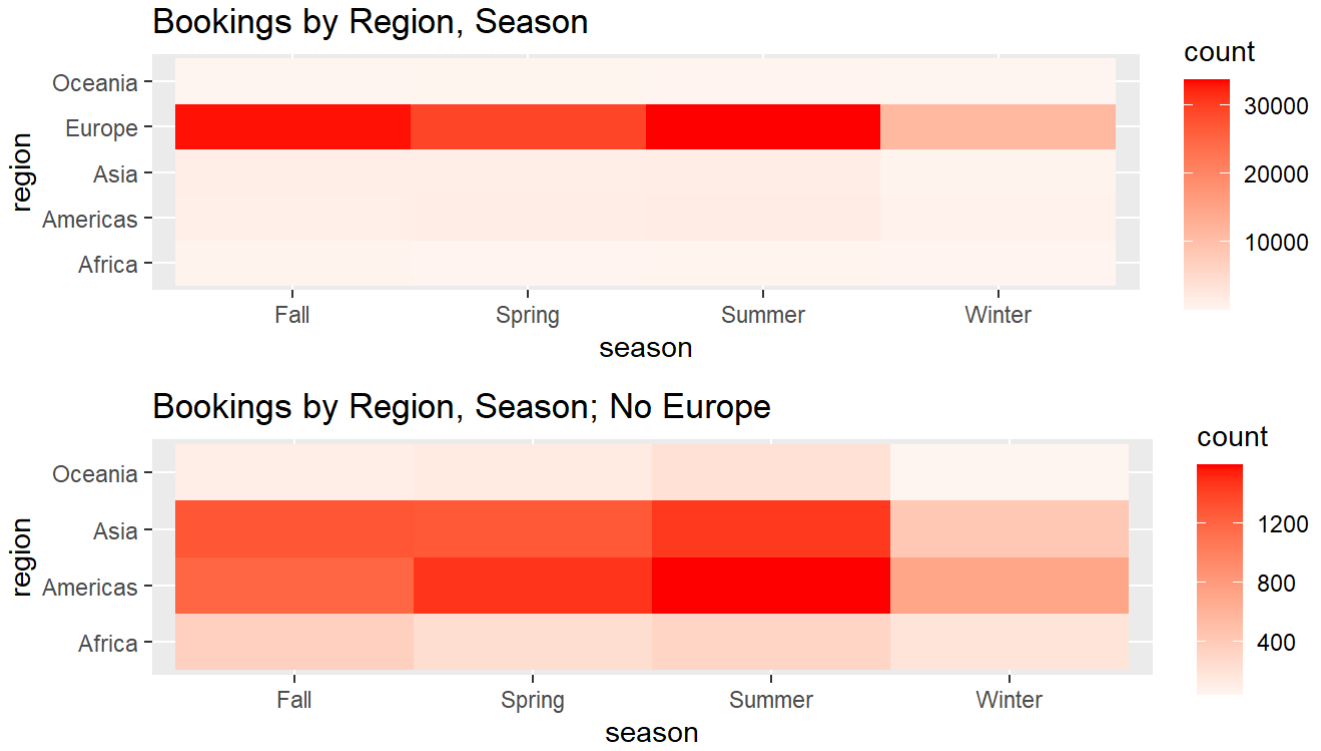
```
## [1] "TPR: 0.277812952243126"
```

```
## [1] "TNR: 0.886863741585291"
```

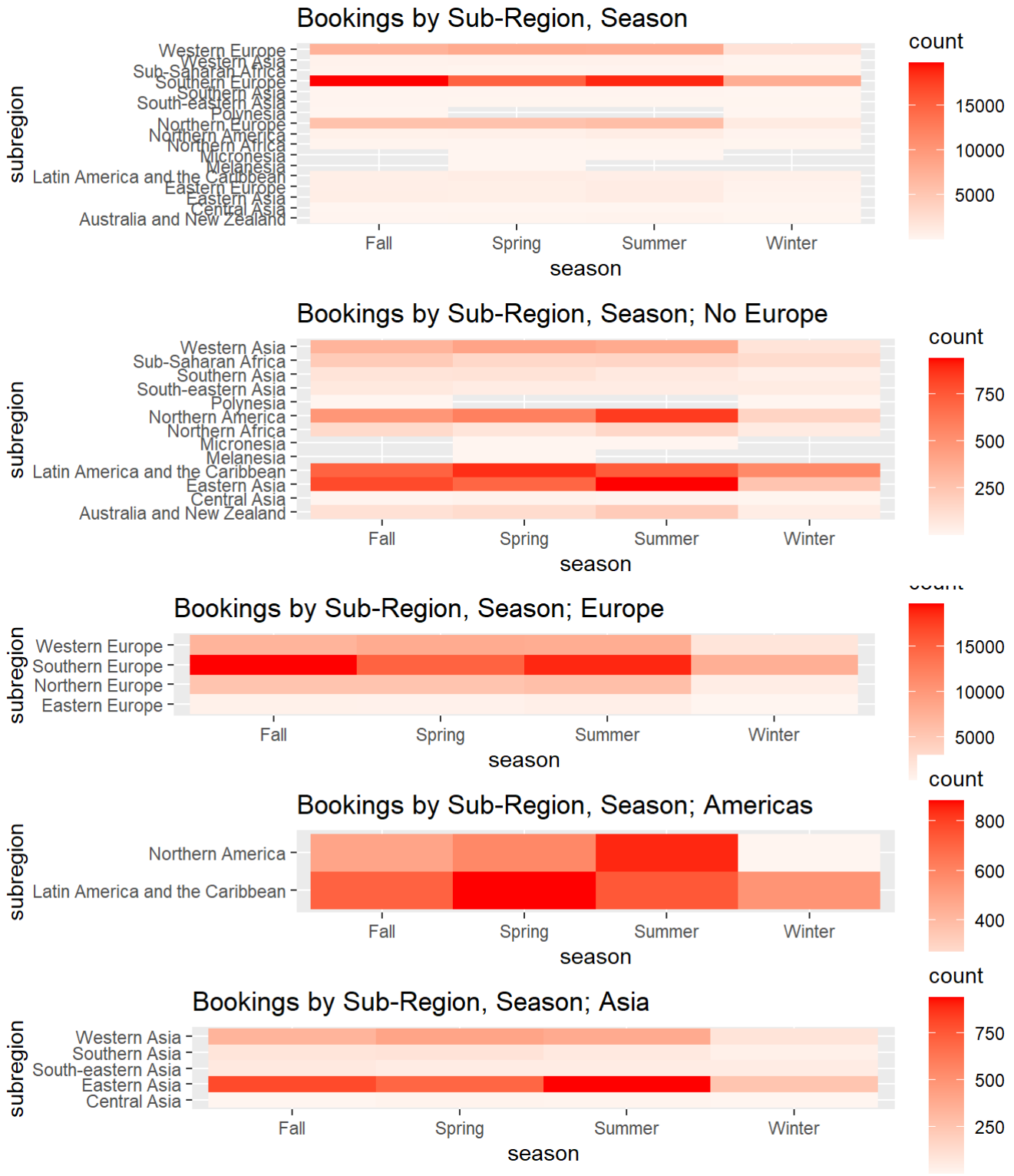
Predicting cancellation from `lead_time` is surprisingly effective, with an accuracy of 66.1%, TPR of 27.8%, and TNR of 88.7%. This performance reveals that `lead_time` is a strong predictor of cancellation, since there are significant quantities of predictions for both categories.

---

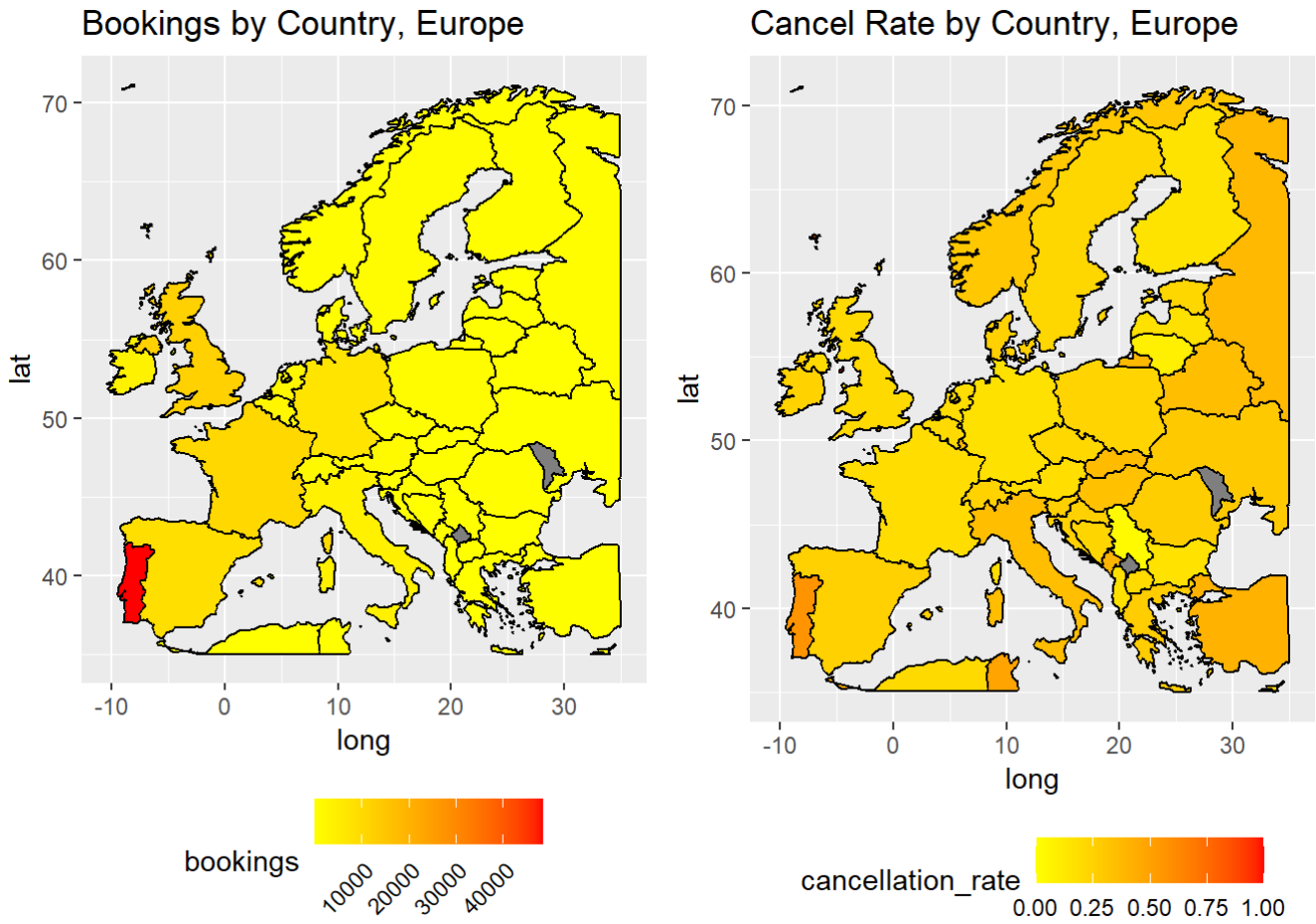
3. What continents/region are people booking from? Which season do the most people travel from each continent/region?



According to the first heat map, Europe is the region which has the most bookings by a large amount. The second heat map reveals that the Americas and Asia are the regions with the second and third most bookings respectively. Summer appears to be the most popular season in each region, while fall and spring have similar amounts of overall bookings. Winter appears to be the least popular season regardless of region.



Examining the bookings by sub-regions, we can see that within Europe, most bookings come from Southern Europe. Since the hotels are located in Portugal, this seems to indicate that many of the booked stays may be domestic trips. The heat map of European sub-regions reveals that Western and Northern Europe have similar amounts of trips booked, while Eastern Europe has relatively few trips overall. In the Americas, the two subregions appear to have similar overall bookings, though North America's bookings are heavily concentrated in the summer. Latin America and the Caribbean more consistent bookings throughout the year, with spring being the most popular. In Asia, the vast majority of bookings come from Eastern Asia, with other three regions booking in similar amounts.

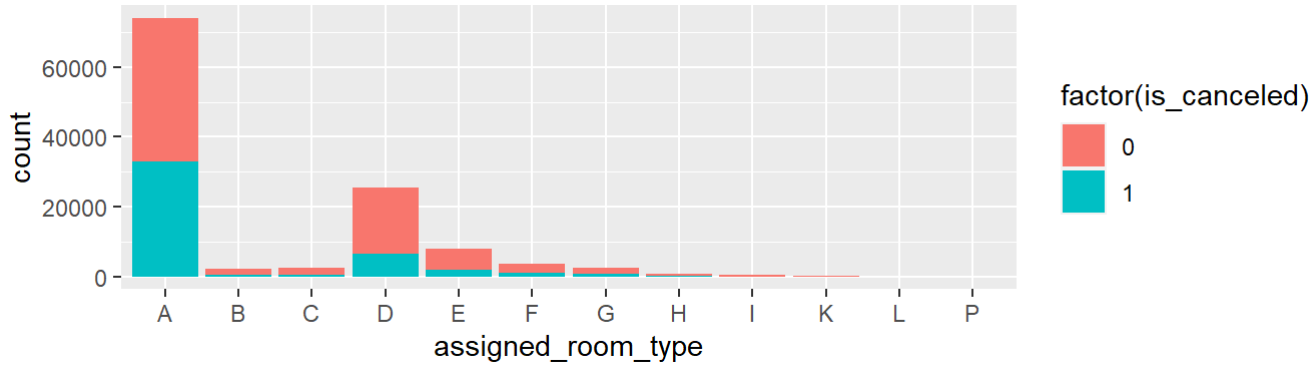


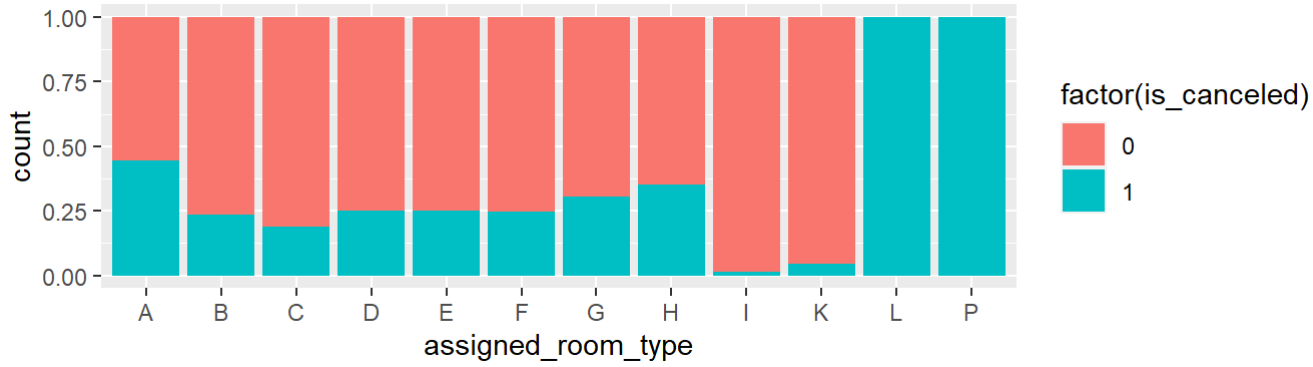
The map above reveals the booking amounts of each European country. Just as revealed by the heat maps of subregions, Southern Europe has the largest amount of bookings. Moving more north and east follows the expected trend of decreased bookings. This trend makes sense since the different subregions are reasonable proxies for the relative wealth of the countries within those subregions. Western and Southern Europe tend to contain the wealthiest countries, while Eastern Europe tends to contain the poorest countries in Europe. Southern Europe's large amount of bookings can likely be attributed to the ease of access since the hotels are located in Portugal. The cancellation rates appear to follow a reverse trend compared to the booking amounts of each subregion. Eastern Europe appears to have higher cancellation rates than Western and Northern Europe. Portugal has a noticeably high cancellation rate, though this is likely due to the high amount of bookings coming from Portugal.

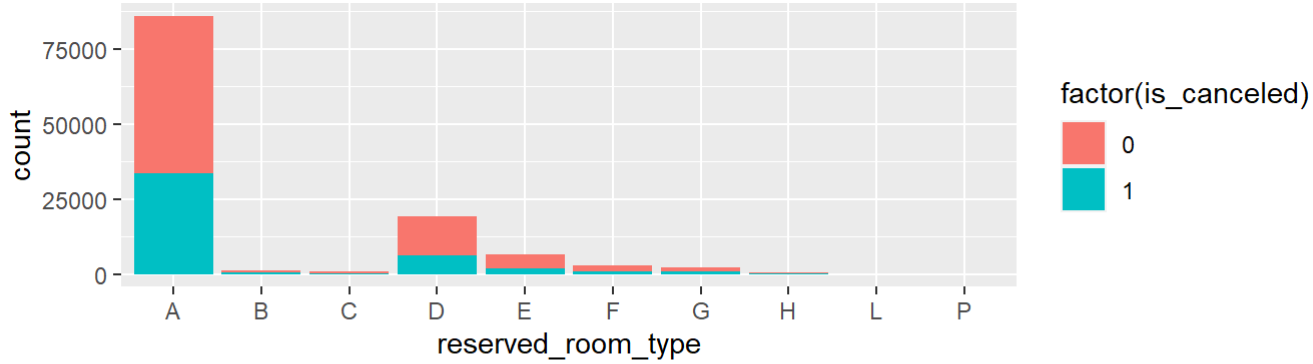
region	num_bookings	cancel_rate
<chr>	<int>	<dbl>
Africa	1114	0.4299820
Americas	4982	0.2980731
Asia	4468	0.3278872
Europe	107826	0.3766439
Oceania	507	0.2268245
NA	493	0.1379310

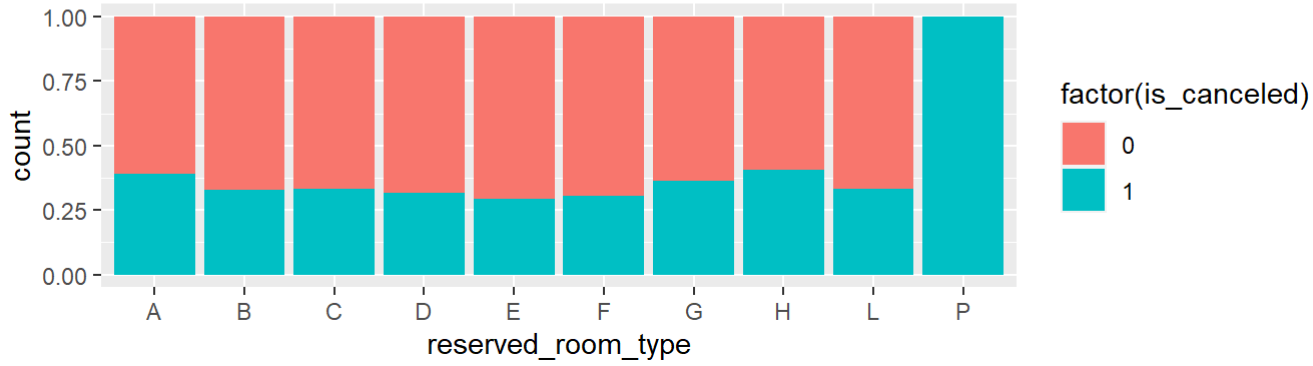
6 rows

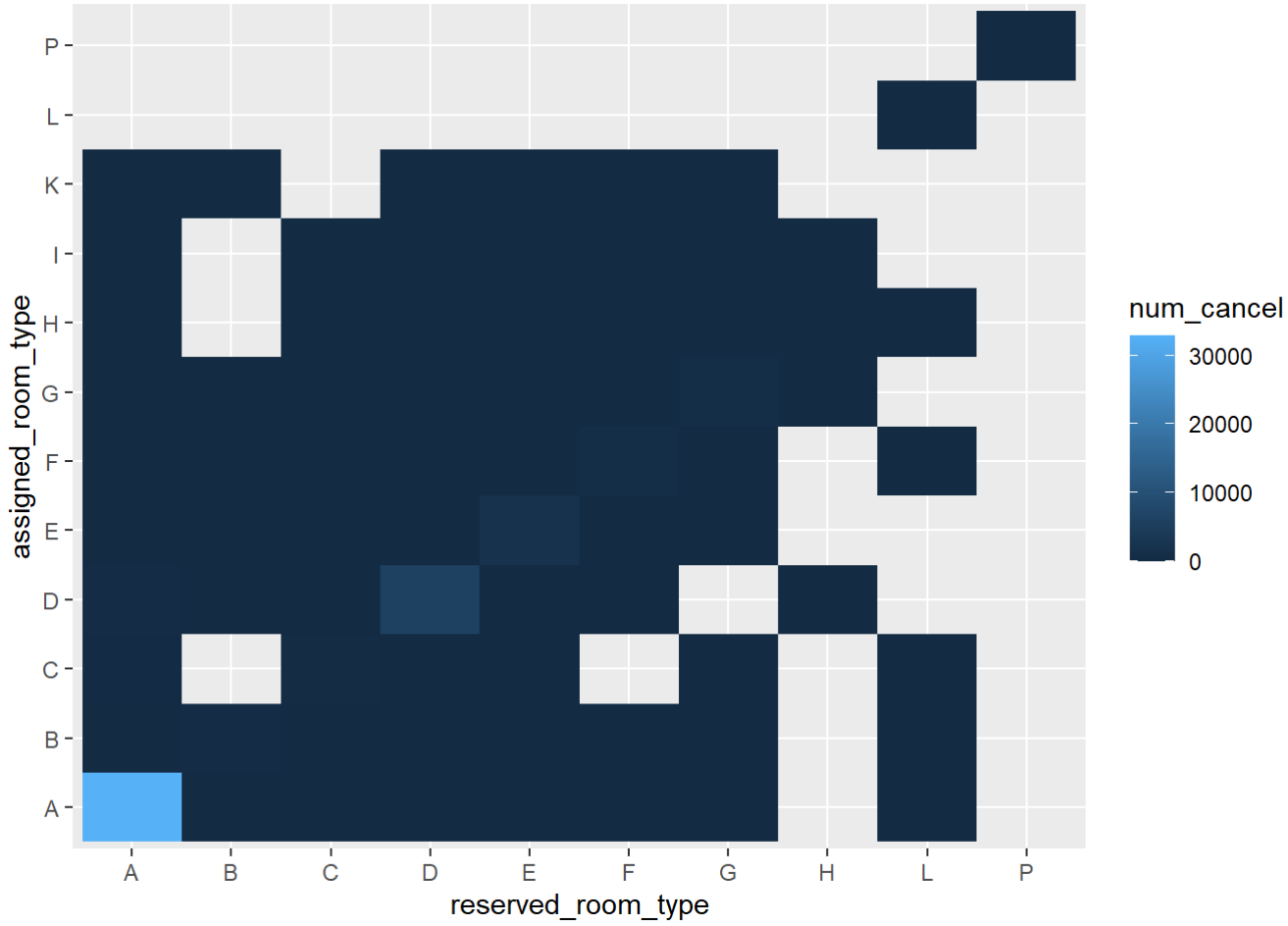
4. How does the type of room correlate with cancellation?











The cancellation rate based on room type is fairly consistent, so there is likely no relationship between cancelling and room type. There is also a very large imbalance in the room types represented, with type A representing a very large majority of the assigned and reserved rooms. This imbalance in category sizes exacerbates the weakness of any meaningful relationship that could exist with cancellation.

```
##        predict_assign_room
##         FALSE  TRUE
##   FALSE 75166     0
##   TRUE  44211    13
```

```
## [1] "Assigned Room Evaluation"
```

```
## [1] "Accuracy: 0.629692604070693"
```

```
## [1] "TPR: 0.000293958031837916"
```

```
## [1] "TNR: 1"
```

```
##        predict_reserve_room
##         FALSE  TRUE
##   FALSE 75166     0
##   TRUE  44212    12
```

```
## [1] "Reserved Room Evaluation"
```

```
## [1] "Accuracy: 0.629684228159812"
```

```
## [1] "TPR: 0.000271345875542692"
```

```
## [1] "TNR: 1"
```

```
##        predict_room
##         FALSE  TRUE
##   FALSE 74644   522
##   TRUE  44192    32
```

```
## [1] "Assigned & Reserved Room Evaluation"
```

```
## [1] "Accuracy: 0.625479520897898"
```

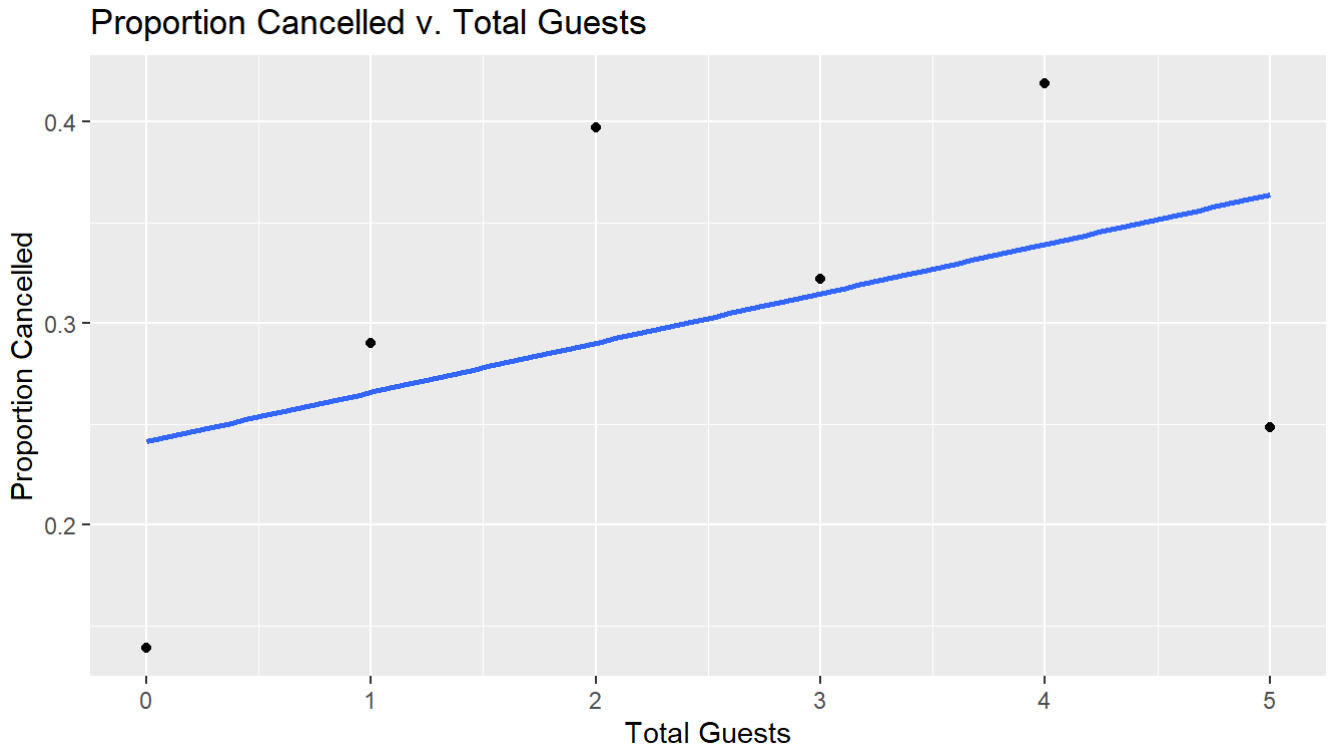
```
## [1] "TPR: 0.000723589001447178"
```

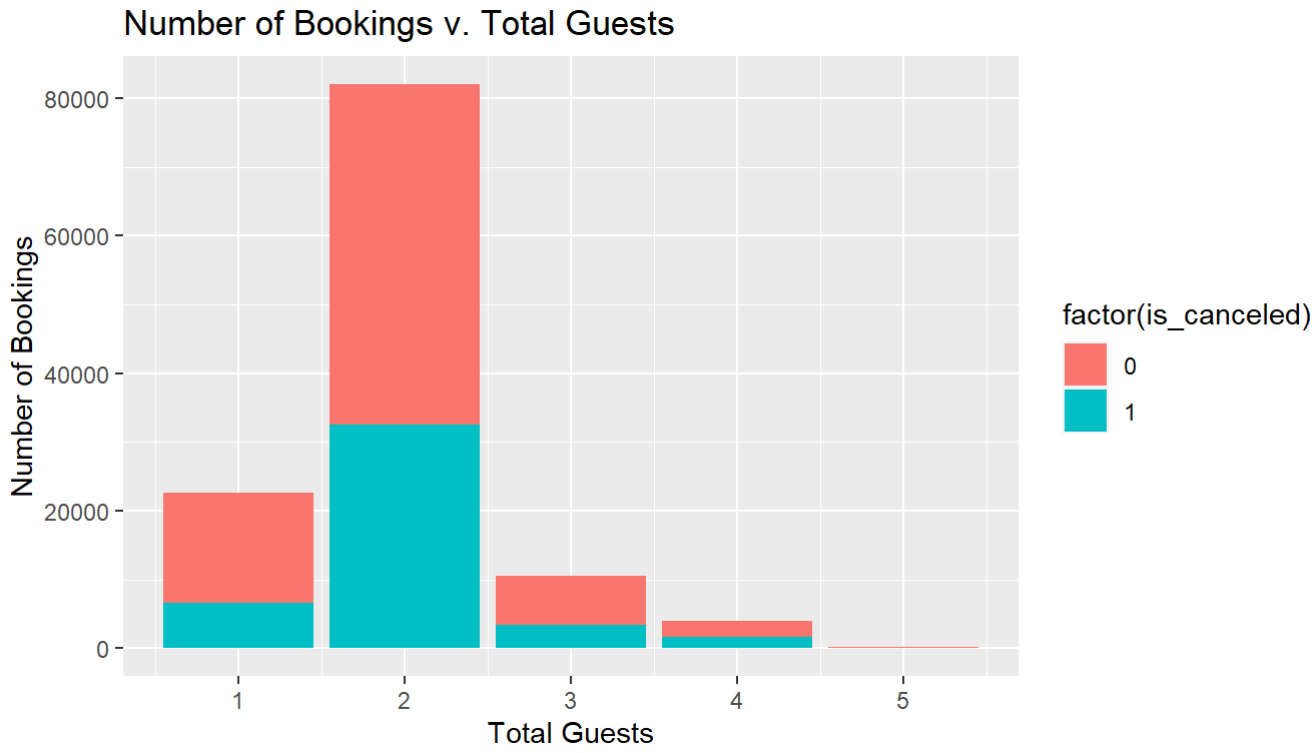
```
## [1] "TNR: 0.993055370779342"
```

The results of the logistic regressions between cancellation and room type, whether reserved or assigned, indicate that room type has no meaningful relationship with cancellation. Each of the three logistic regressions have an accuracy of 62%, which is only slightly better than guessing if a guest will cancel. Additionally, the TNR is effectively 1, while the TPR is effectively 0. Since these two metrics are at extremes, the model must be guessing since the vast majority of the predictions are that a booking will not be cancelled. This means that the accuracy of the model can be attributed to the imbalance in the number of cancellations and the number of non-cancellations. If the cancellation rade were 50%, the model's accuracy would be 50%.

---

5. Does the number of guests affect the likelihood of a booking being cancelled?





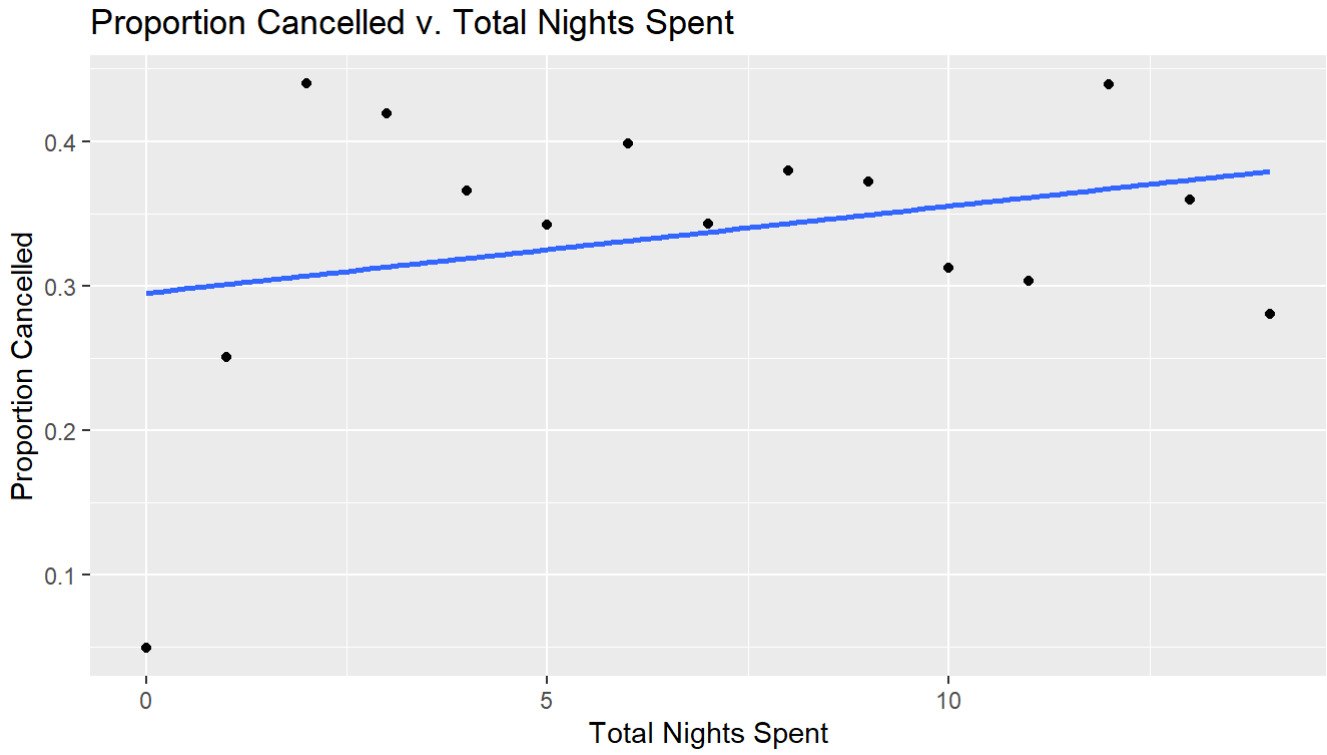
```
##
## Call:
## glm(formula = proportion ~ total_guests, data = result_guests)
##
## Deviance Residuals:
##         1          2          3          4          5          6
## -0.102379   0.024531   0.106704   0.006972   0.079711  -0.115540
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.24127    0.07436   3.245   0.0315 *
## total_guests  0.02449    0.02456   0.997   0.3751
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.01055519)
##
##     Null deviance: 0.052716  on 5  degrees of freedom
## Residual deviance: 0.042221  on 4  degrees of freedom
## AIC: -6.7124
##
## Number of Fisher Scoring iterations: 2
```

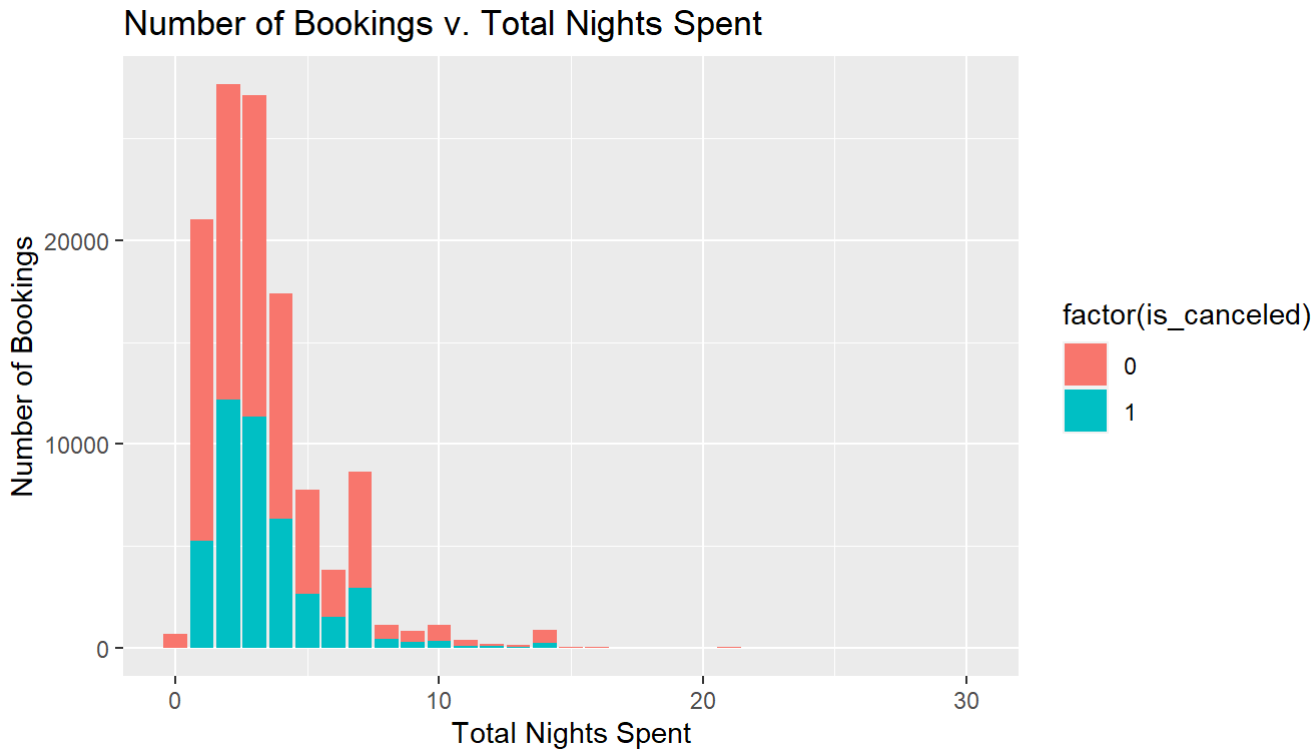
```
##
## Call:
## glm(formula = is_canceled ~ total_guests, family = binomial,
##     data = hotel_bookings)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5853  -0.9634  -0.9085   1.4078   1.5379
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.816539   0.018874  -43.26   <2e-16 ***
## total_guests  0.144842   0.009035   16.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 157390  on 119385  degrees of freedom
## Residual deviance: 157120  on 119384  degrees of freedom
##   (4 observations deleted due to missingness)
## AIC: 157124
##
## Number of Fisher Scoring iterations: 4
```

The total number of guests does not have a significant relationship with the cancellation rate, but it does have a relationship indicating if a particular booking will be cancelled. The majority of bookings were made for 2 guests, and such bookings had a higher proportion of cancellations. This may be due to the imbalance in number of guests, so number of guests may not be a very reliable indicator of cancellation.

---

6. Do cancellations occur in greater proportion the more days are booked?





```
##
## Call:
## glm(formula = proportion ~ total_nights, data = result)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.24586  -0.04628   0.01718   0.05694   0.13315
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.294811   0.047472   6.210 3.17e-05 ***
## total_nights 0.006023   0.005771   1.044    0.316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.00932521)
##
##     Null deviance: 0.13139  on 14  degrees of freedom
## Residual deviance: 0.12123  on 13  degrees of freedom
## AIC: -23.704
##
## Number of Fisher Scoring iterations: 2
```

```
##
## Call:
## glm(formula = is_canceled ~ total_nights, family = binomial,
##     data = hotel_bookings)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3521  -0.9596  -0.9487   1.4058   1.4312
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.579334   0.009995 -57.962  < 2e-16 ***
## total_nights  0.014219   0.002319   6.133 8.65e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 157398  on 119389  degrees of freedom
## Residual deviance: 157361  on 119388  degrees of freedom
## AIC: 157365
##
## Number of Fisher Scoring iterations: 4
```

The total number of nights does not have a significant relationship with the cancellation rate, but it does have a relationship with determining if a particular booking will be cancelled. Although the proportion of cancellations appears to increase with the number of total nights spent, the increase in cancellation rate can be explained by the small sample size. Most bookings were for 7 days or less, so the higher cancellation rate for longer trips is due to low sample size.

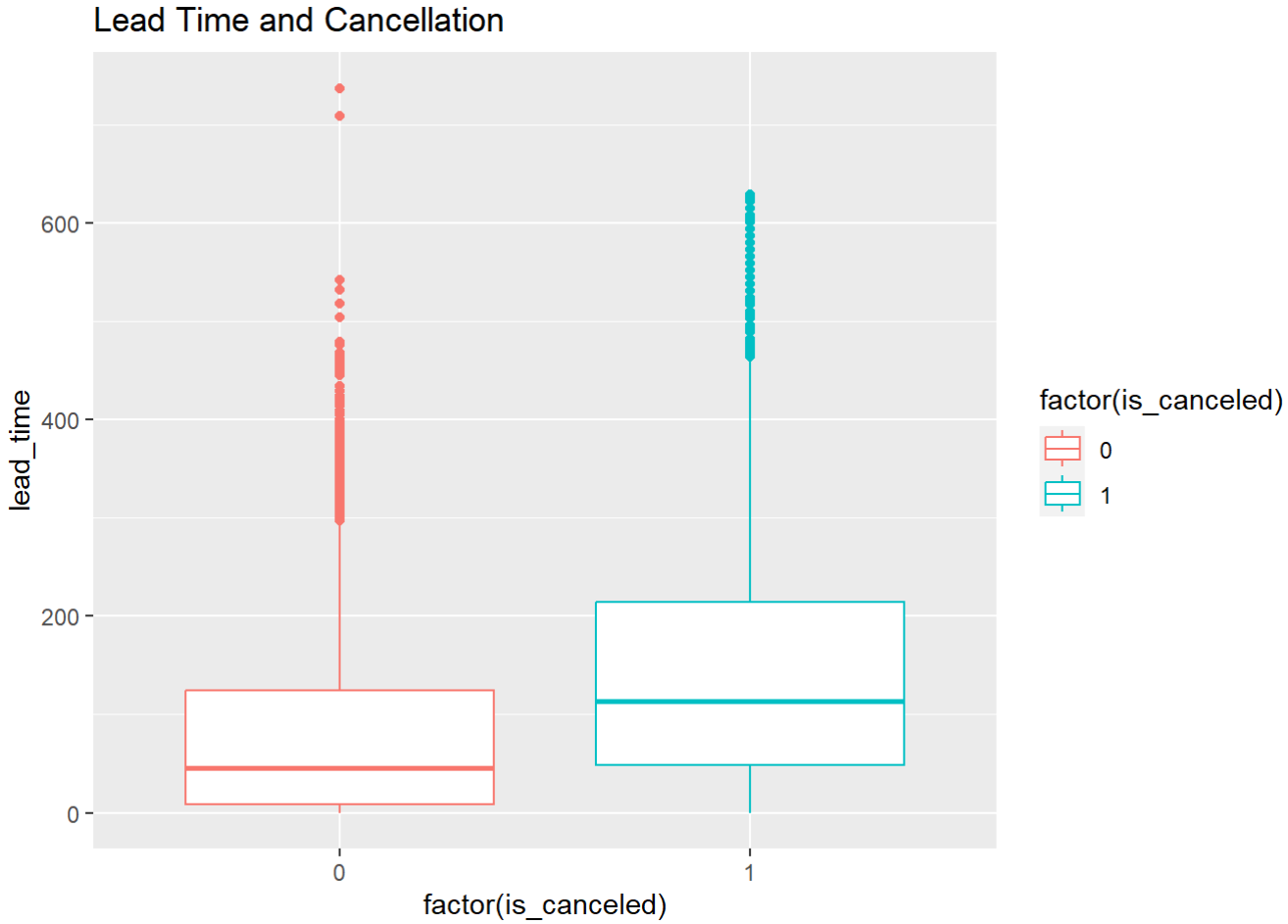
7. Are business trips more likely to have hotel cancellations?

distribution_channel <chr>	num_canceled <dbl>	total <int>	prop_canceled <dbl>
TA/TO	40152	97870	0.4102585
Direct	2557	14645	0.1745988
Corporate	1474	6677	0.2207578
GDS	37	193	0.1917098
Undefined	4	5	0.8000000

5 rows

Business trips appear to have a lower proportion of cancellations, but it is possible that the lower proportion is due to small sample size. Only about 6500 trips were business trips, which is only small portion of the overall data.

8. Does the days booked in advance affect hotel cancellation?



The boxplots show that the median `lead_time` differs for trips based on whether they were canceled or not. This indicates that `lead_time` likely has a significant relationship with cancellation.

Predictive Models

Logistic Regression

```
logit = glm(f, binomial(), train)
prob = predict(logit, type = "response", train)
cancel_predictions_train = tibble(canceled = as.logical(train$is_canceled),
                                  prob = prob, predict = prob > 0.5)
logit.train.confusion.matrix = table(cancel_predictions_train$canceled,
                                     cancel_predictions_train$predict)
prob = predict(logit, type = "response", test)
cancel_predictions_test = tibble(canceled = as.logical(test$is_canceled),
                                 prob = prob, predict = prob > 0.5)
logit.test.confusion.matrix = table(cancel_predictions_test$canceled,
                                    cancel_predictions_test$predict)
```

```
## [1] "Training Set Evaluation"
```

```
##
##         FALSE  TRUE
##   FALSE 56205  3510
##   TRUE  15173 20222
```

```
## [1] "Accuracy: 0.800182176061647"
```

```
## [1] "TPR: 0.571323633281537"
```

```
## [1] "TNR: 0.941220798794273"
```

```
## [1] "Testing Set Evaluation"
```

```
##
##         FALSE  TRUE
##   FALSE 14222   804
##   TRUE   3646  5115
```

```
## [1] "Accuracy: 0.809824943462602"
```

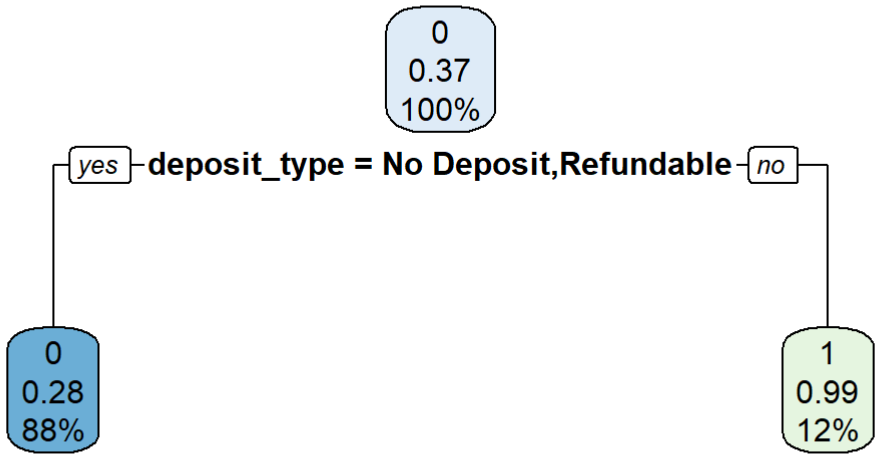
```
## [1] "TPR: 0.583837461477"
```

```
## [1] "TNR: 0.946492745907094"
```

Performing logistic regression yields fairly high model accuracy, though the TPR appears to suffer in exchange for a very high TNR. This indicates that the model can accurately identify guests that will not cancel, but may still be guessing when guests actually cancel. The training and testing sets yielded similar performance, so we know that the model is representative of the overall data, rather than a very specific subset.

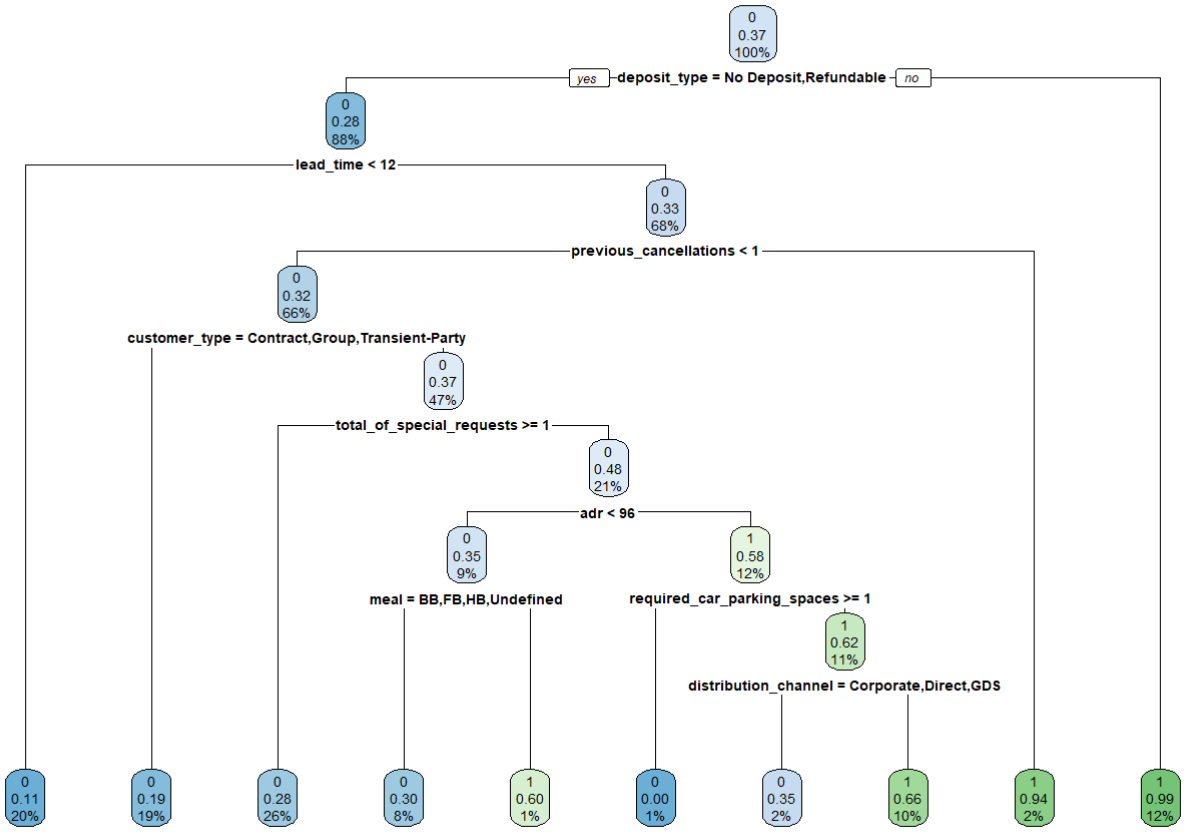
Decision Tree

```
training_model <- rpart(is_canceled ~ .,
                        data=features_1,
                        method="class",
                        control=rpart.control(cp=0.3))

rpart.plot(training_model)
```



`rpart` organizes the features by significance and the number of features used is decided by the complexity. From this low complexity decision tree we can see that `deposit_type` is the most significant factor for predicting hotel cancellations.

```
max_cp_model <- rpart(is_canceled ~ .,
                      data=features_1,
                      method="class",
                      control=rpart.control(cp=0.005))

rpart.plot(max_cp_model)
```



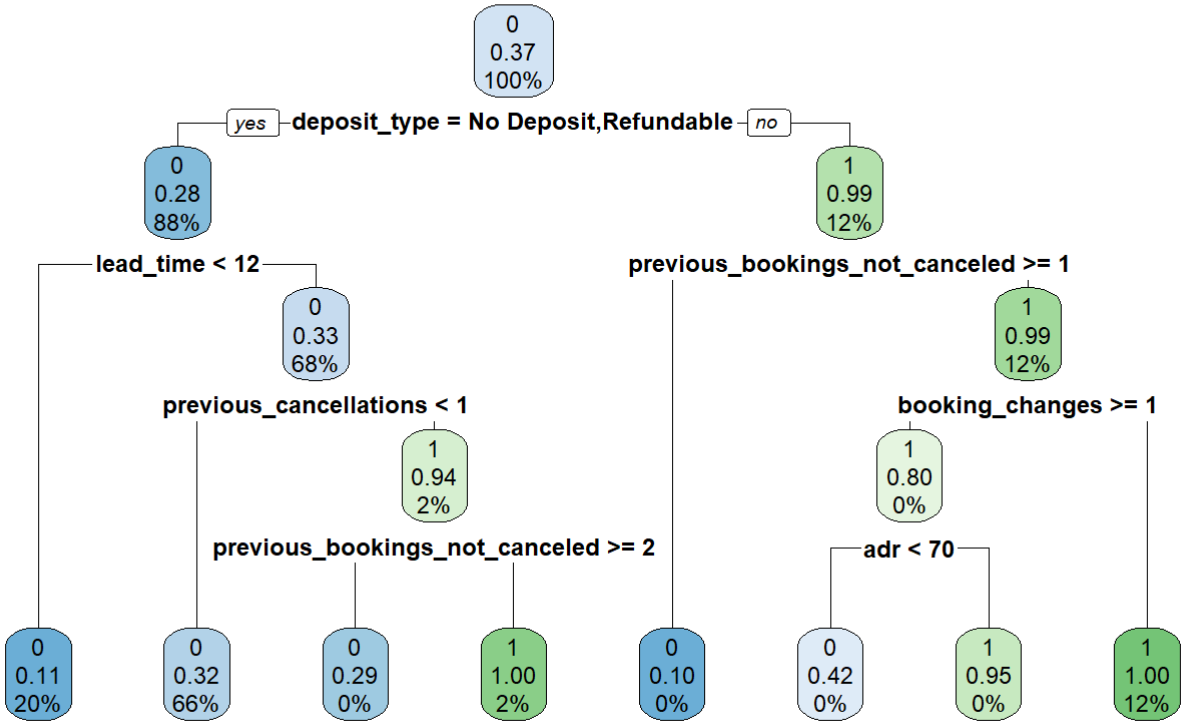
A

high complexity decision tree is very likely to be overfit, and as a result, may have many irrelevant features that don't contribute much to prediction. To avoid overfitting, high complexity trees should be pruned to only include the most relevant features.

```
#classification tree probability
test$ct_pred_prob <- predict(training_model,test)[,2]
test$ct_pred_class <- predict(training_model,test,type="class")

table(test$is_canceled == test$ct_pred_class)
```

```
##
## FALSE  TRUE
##  7446 22401
```

Unpruned Full Decision Tree



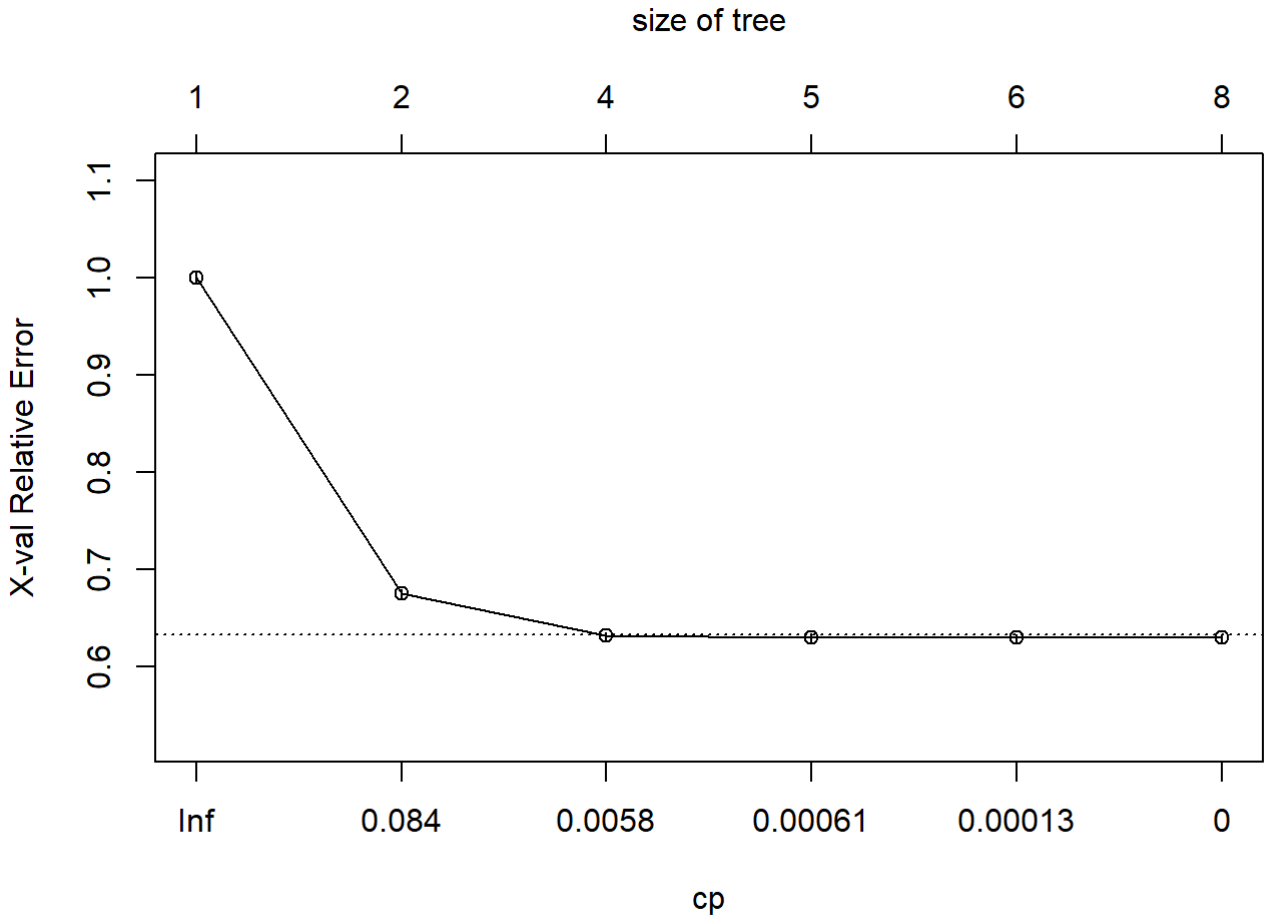

```
# which features are most important to the decision tree
full_tree$variable.importance
```

```
##                   deposit_type         previous_cancellations
##                   9661.6455910                   1806.7605928
##                      lead_time previous_bookings_not_canceled
##                   1788.9052238                    247.7180199
##              is_repeated_guest           distribution_channel
##                     99.6428830                     67.1783725
##                            adr    required_car_parking_spaces
##                     43.3730183                     28.0199961
##      total_of_special_requests                booking_changes
##                     14.0099980                     12.4309854
##                          hotel                         adults
##                      5.1185128                      0.7755721
```

```
printcp(full_tree)
```

```
##
## Classification tree:
## rpart(formula = is_canceled ~ ., data = features_1, method = "class",
##     control = rpart.control(cp = 0, maxdepth = 4))
##
## Variables actually used in tree construction:
## [1] adr                           booking_changes
## [3] deposit_type                  lead_time
## [5] previous_bookings_not_canceled previous_cancellations
##
## Root node error: 33158/89543 = 0.3703
##
## n= 89543
##
##           CP nsplit rel error  xerror      xstd
## 1 3.2514e-01      0   1.00000 1.00000 0.0043578
## 2 2.1895e-02      1   0.67486 0.67486 0.0039073
## 3 1.5381e-03      3   0.63107 0.63140 0.0038197
## 4 2.4127e-04      4   0.62953 0.62980 0.0038163
## 5 7.5397e-05      5   0.62929 0.62938 0.0038154
## 6 0.0000e+00      7   0.62914 0.62953 0.0038157
```

```
# plot complexity parameter
plotcp(full_tree)
```

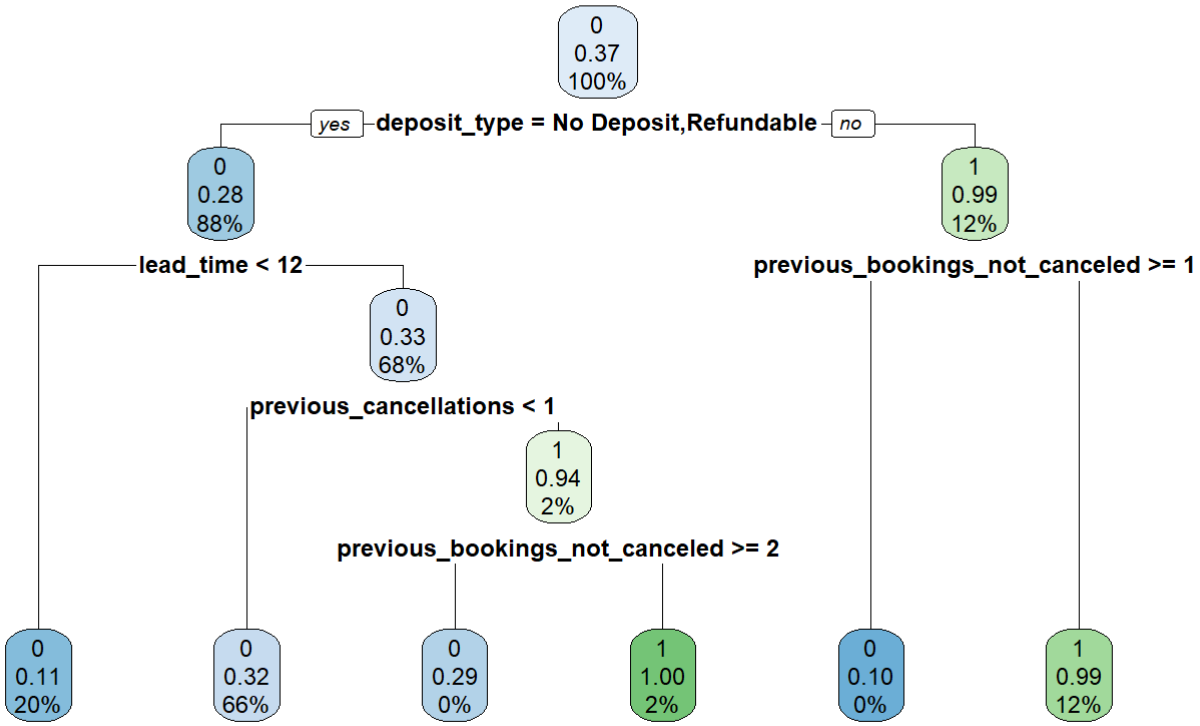


`cpplot()` does cross validation for different complexity levels and plots the cross validation error on the y-axis.

```
##           CP       nsplit    rel error       xerror         xstd
## 7.539659e-05 5.000000e+00 6.292901e-01 6.293805e-01 3.815424e-03
```

```
##           CP
## 7.539659e-05
```

Pruned Decision Tree



After determining that a tree with complexity of 4 is the most effective tree, we prune the tree so that only 4 decisions are made. The decisions made by the pruned tree will only reflect the 4 most important features.

```
##    tree.pred
##         0     1
##   0 18758    23
##   1  6874  4192
```

```
## [1] "Accuracy: 0.768921499648206"
```

```
## [1] "TPR: 0.378818001084403"
```

```
## [1] "TNR: 0.99877535807465"
```

The decision tree has an accuracy of 76.9%, a TPR of 37.9% and a TNR of 99.8%. Overall this performance is similar to that of the logistic regression, though the TPR is noticeably smaller than the logistic regression's 57%. However, the logistic regression used many more predictors to achieve similar performance, so the decision tree likely can likely be more generalized since the logistic regression could be overfit.

KNN

```
knn_fit <- train(is_canceled ~ deposit_type + lead_time + previous_cancellations +
                 previous_bookings_not_canceled + is_repeated_guest + distribution_channel +
                     adr, data = train, method = "knn",
                 trControl = trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10)
```

```
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined
```

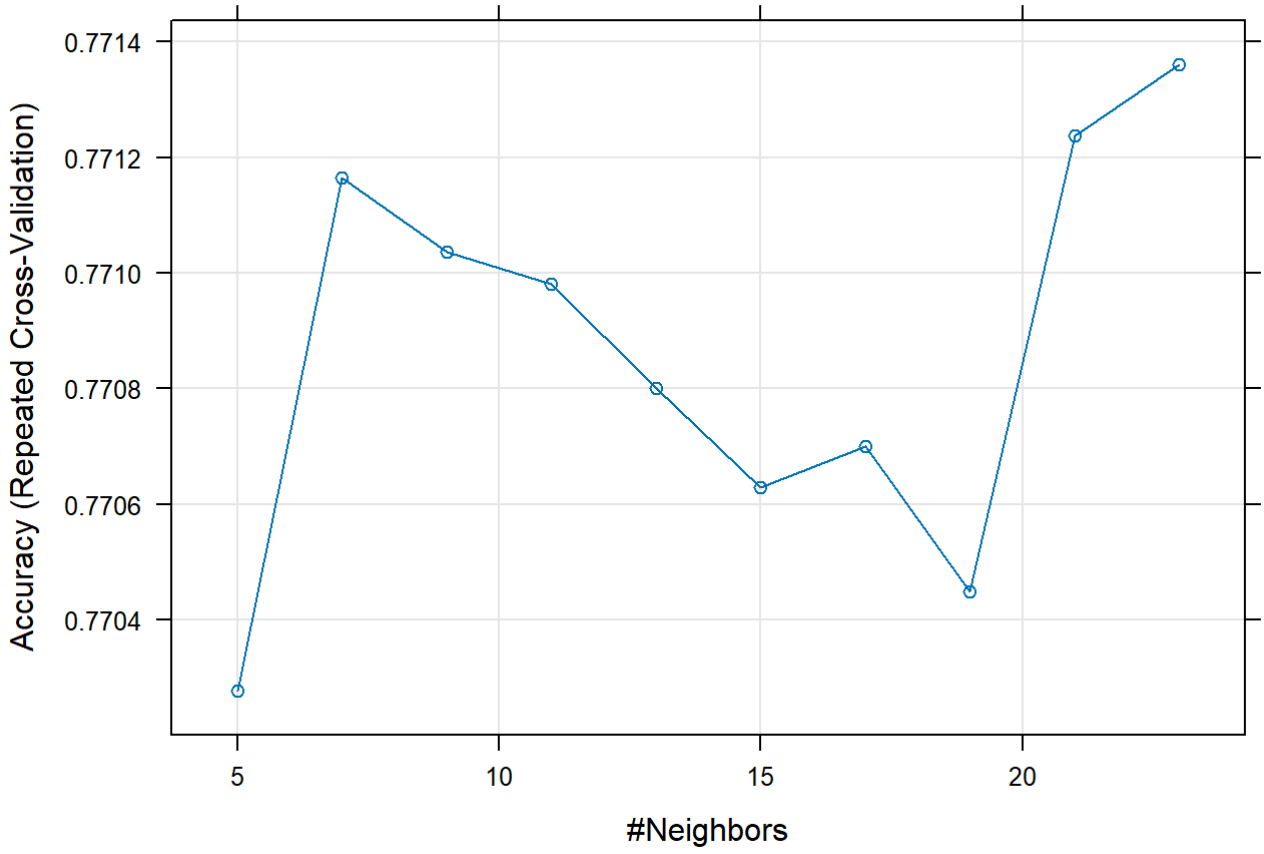
```
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined
```

```
## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19, uniqueCut =
## 10, : These variables have zero variances: distribution_channelUndefined
```

```
knn_fit
```

```
## k-Nearest Neighbors
##
## 83226 samples
##     7 predictor
##     2 classes: '0', '1'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 74904, 74904, 74903, 74903, 74903, 74904, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.7702761  0.4899290
##    7  0.7711653  0.4881829
##    9  0.7710371  0.4845797
##   11  0.7709810  0.4816011
##   13  0.7708008  0.4789774
##   15  0.7706285  0.4766326
##   17  0.7707006  0.4750568
##   19  0.7704483  0.4731532
##   21  0.7712373  0.4736476
##   23  0.7713615  0.4726596
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 23.
```

```
# accuracy vs k-folds
plot(knn_fit)
```



From the plot and printout, we can see that $k = 11$ yields the most accurate model. However, since the classes are imbalanced, κ may be a more appropriate metric. Using κ the best model is actually $k = 5$ though the difference in performance is not very significant.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 20491  6432
##          1  1819  6925
##
##                Accuracy : 0.7687
##                  95% CI : (0.7643, 0.773)
##     No Information Rate : 0.6255
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4695
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9185
##             Specificity : 0.5185
##          Pos Pred Value : 0.7611
##          Neg Pred Value : 0.7920
##              Prevalence : 0.6255
##          Detection Rate : 0.5745
##    Detection Prevalence : 0.7548
##       Balanced Accuracy : 0.7185
##
##        'Positive' Class : 0
##
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 48802 13902
##          1  3629 16893
##
##                Accuracy : 0.7894
##                  95% CI : (0.7866, 0.7921)
##     No Information Rate : 0.63
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5148
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9308
##             Specificity : 0.5486
##          Pos Pred Value : 0.7783
##          Neg Pred Value : 0.8232
##              Prevalence : 0.6300
##          Detection Rate : 0.5864
##    Detection Prevalence : 0.7534
##       Balanced Accuracy : 0.7397
##
##        'Positive' Class : 0
##
```

Findings and Conclusions

From our models, we learned that the most important predictors of cancellation were variables related to how a booking is paid for as well as a guest`s previous history of booking at the hotel. Overall, each of the three models we used performed pretty similarly. The decision tree is the most easily interpretable model since its design mimics human decision-making processes. Although the TPR is lower for the tree, its simplicity makes up for the loss in ability to distinguish which bookings will actually cancel. The tree is able to make predictions using a maximum of 4 variables, but the logistic regression requires many more predictors for slightly better performance. Similarly, the tree can be built much more quickly than the KNN model, which makes it useful for quick insight. The KNN model is the most accurate model and has the highest TPR, but comes at the cost of incredibly long run times due to the size of the data. The TPR is also only slightly higher than that of the logistic regression, though the KNN uses only 7 predictors to achieve this performance. We also know that it is actually finding patterns in the data because its κ coefficient is a value near 0.5. κ can take any value between -1 and 1, where negative values indicate worse performance than guessing, and positive values represent better performance than guessing. Unfortunately, its run times may be prohibitively long, so logistic regression is a much more practical method of prediction. Logistic regression is an effective model for predicting cancellations, and seems to be a good compromise between the speed of the decision tree and complexity of KNN.
