STAT167 Group 11 Final Project: Airline Bookings

Amy Lau, Emlyn Zhai, Lindsay Phan, Adelric Low, Brian Uong

Contents

Project Introduction . . . 1
Research Question . . . 1
Additional Research Questions . . . 1
Project Description . . . 2
Dataset . . . 2
Data Exploration and Visualization . . . 4
Classification Models . . . 13
Model Evaluation . . . 19
Limitations . . . 22
Conclusion . . . 24
Authors Contributions . . . 25
Data/Code Availability . . . 26

Project Introduction

The goal of this research project is to analyze an airline booking data set from Kaggle to determine what factors influence airline bookings. Specifically, our objective is to determine which factors passengers value most when purchasing an airline ticket. By identifying these factors, we hope to better understand passengers' decision-making process, which can be valuable for airlines to optimize their services and marketing strategies. Through our analyses, we aim to discover patterns in booking behaviors so that we can see which elements impact the passengers' decision to book a flight.

Research Question

What factors influence airline booking?

Additional Research Questions

- Does stay length have any correlation with purchase lead?
- Does sales channel correlate with purchase lead?
- Is there any correlation between booking origin and flight route?
- Does origin of route affect the flight duration?
- Is trip type correlated with flight day?
- Does wanting extra baggage/preferred seats/in-flight meals affect airline booking?

Project Description

To achieve our goal and answer the research questions, we will perform exploratory data analysis and determine if there is any correlation between the factors in our data set. We will also create two classification models to predict whether an instance is "yes" or "no" for booking_complete. Naive Bayes and Random Forest are suitable for our data set because booking_complete is a binary variable. Random Forest will also allow us to quantify the importance of the variables in our data set and determine which ones are most influential in predicting whether a booking is completed or not. Our models will be evaluated by calculating the classification metrics and performing 10-fold cross-validation for each model.

Dataset

Airlines Booking

```
# install.packages("e1071", dep = TRUE)
# libraries
# install.packages("randomForest")
library(reshape2)
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(tidyr)
library(dplyr)
library(airportr)
library(maps)
```

```
##
## Attaching package: 'maps'
##
## The following object is masked from 'package:purrr':
##
##     map
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ggdendro)
library(vcd)
```

```
## Loading required package: grid
```

```
library(e1071)
library(randomForest)
```

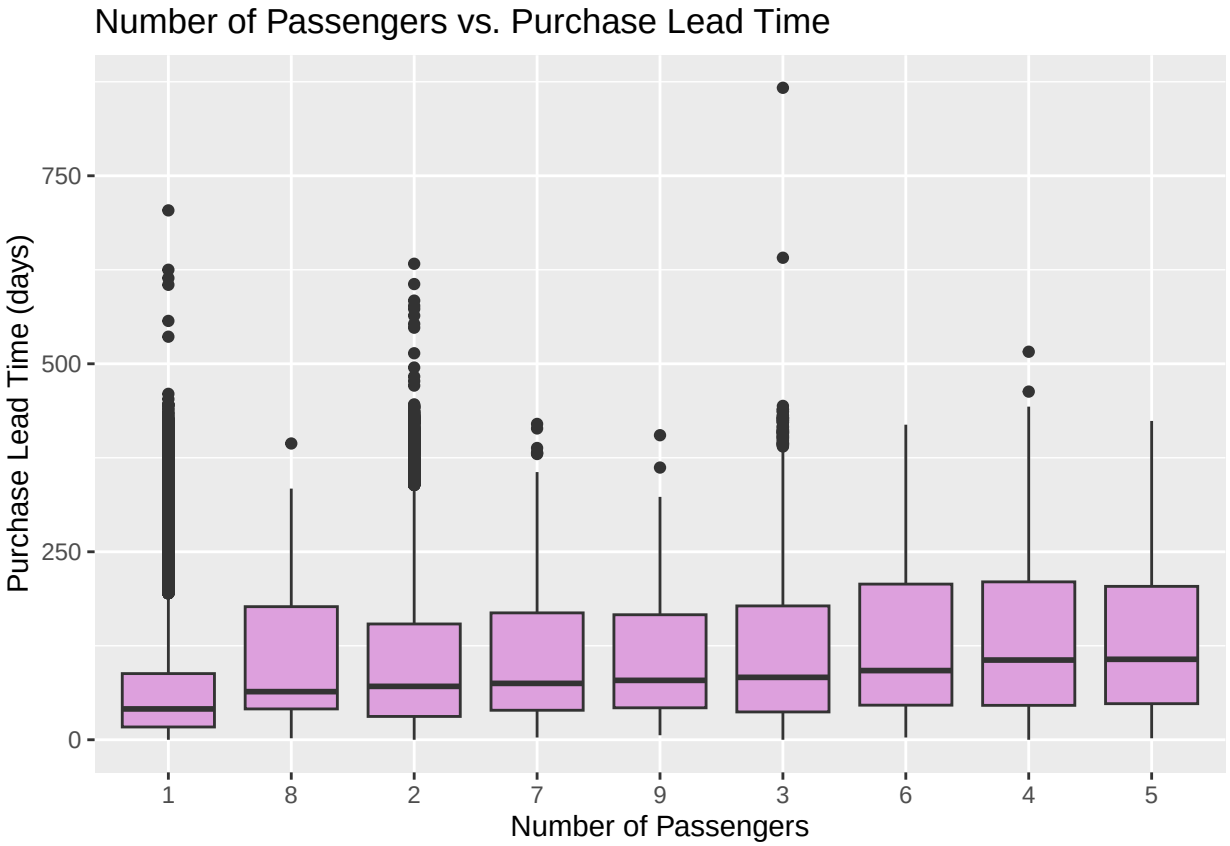
```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:gridExtra':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(boot)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'lattice'
##
## The following object is masked from 'package:boot':
##
##     melanoma
##
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

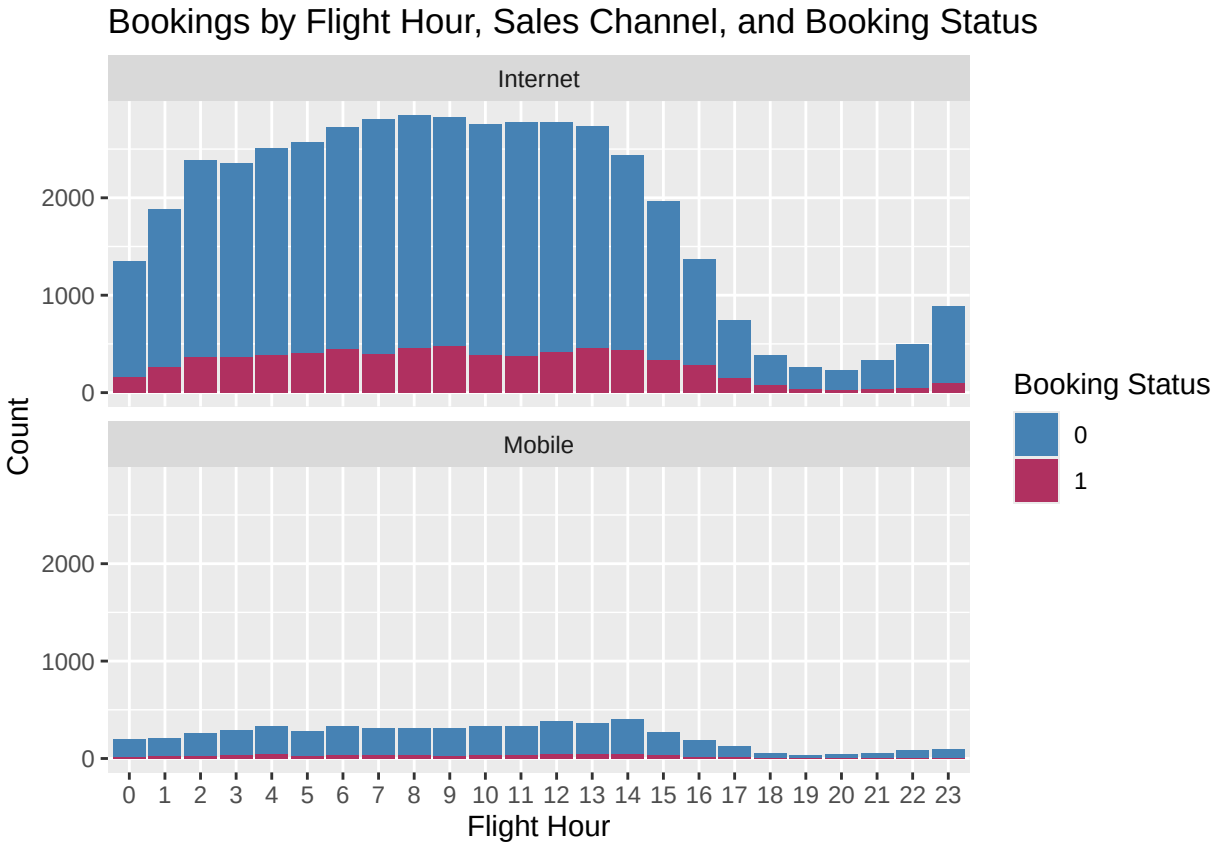
Data Exploration and Visualization

Number of Passengers vs. Purchase Lead Time



From the box plot above, we are comparing the number of passengers vs purchase lead time. We can see from the median of these box plots that a group of 5 passengers has a longer lead time on average. In comparison, we can also see that the lead time is the shortest when there is just one passenger traveling alone.

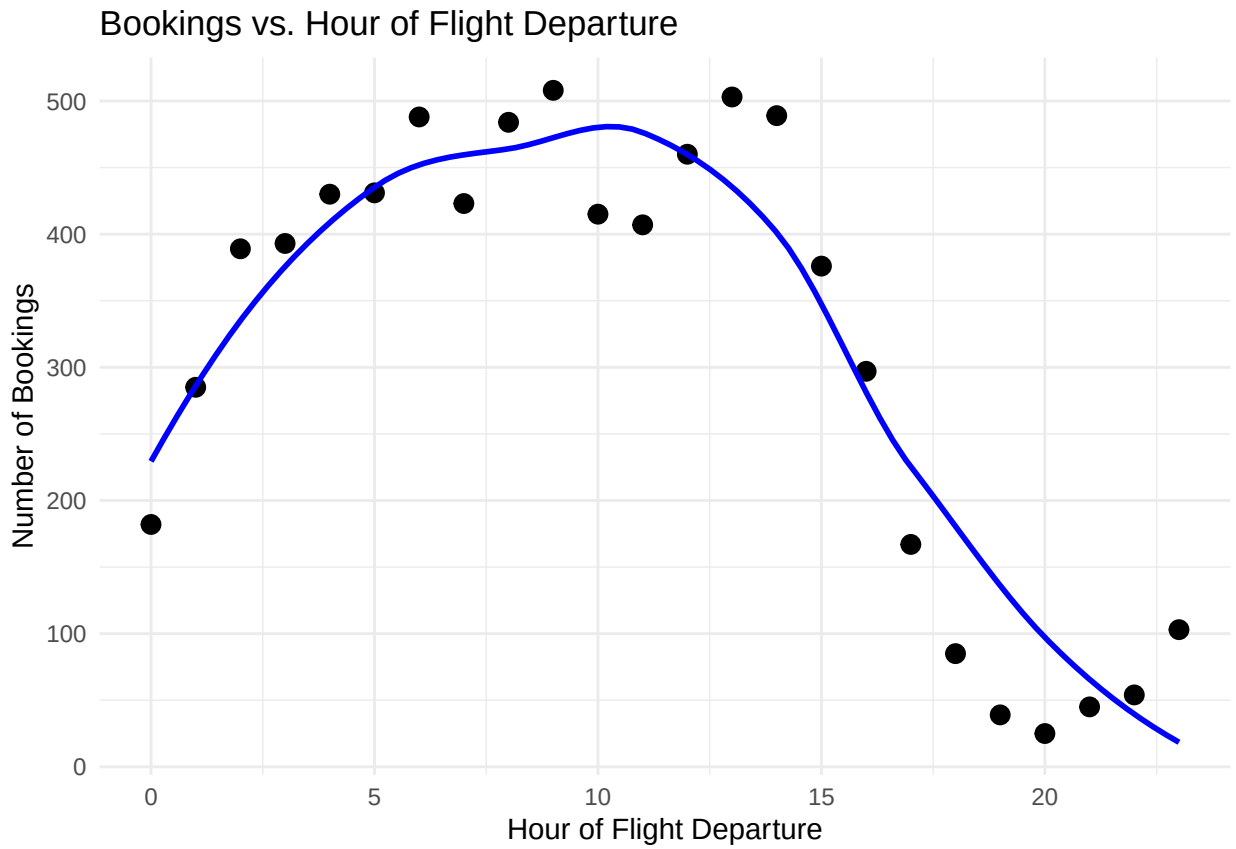
Bookings By Flight Hour, Sales Channel, and Booking Status



For this bar graph, we compared the amount of bookings made for different flight hours across various sales channels. The internet channel is more popular in general, especially in the morning. Peak booking hours for the internet graph are from 7am to 1pm. Bookings at around 9 am had the highest volume sold and the largest amount of not completed bookings being around 8 am. Since the graphs are more right-skewed, we can assume that majority of people tend to book their flights in the morning rather than at night. We can see that passengers prefer to use the Internet over mobile to book their flights since the mobile graph has fewer bookings than the Internet graph.

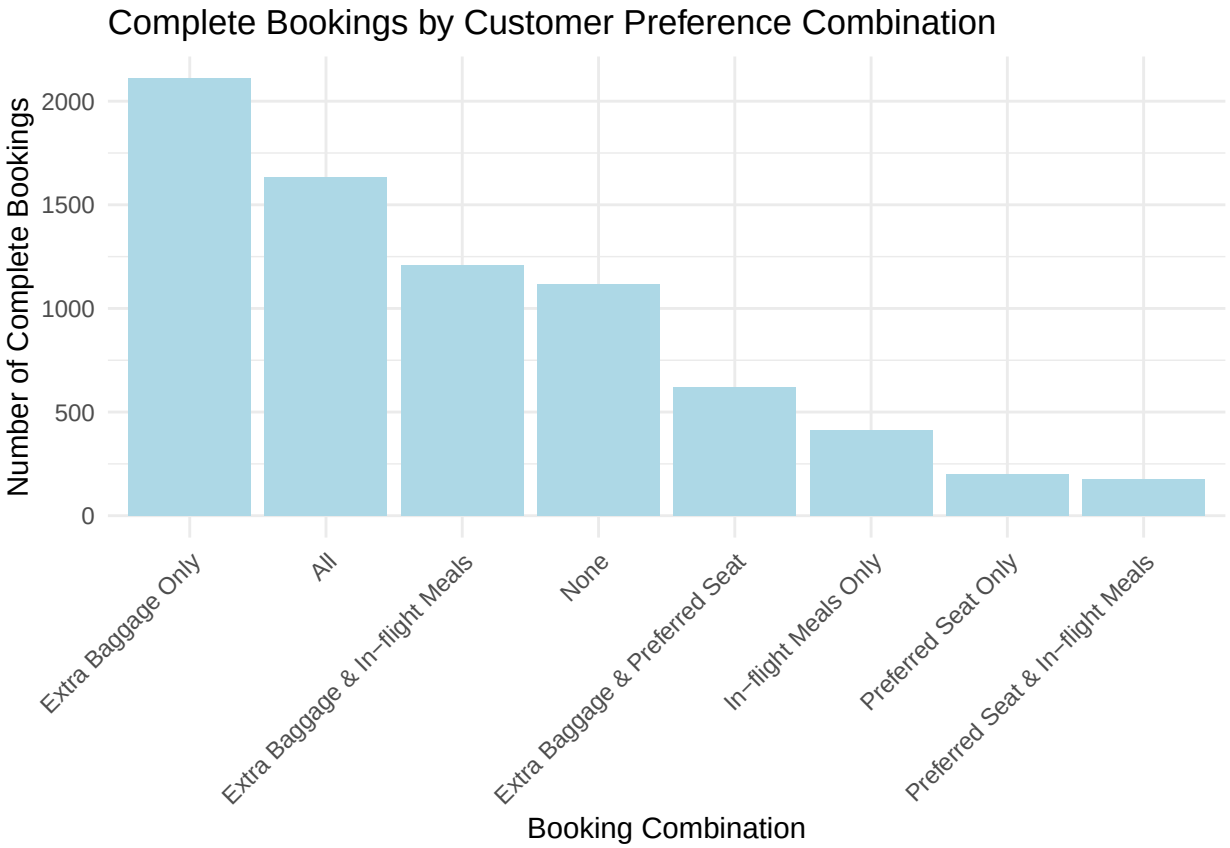
Bookings vs. Hour of Flight Departure

```
## `geom_smooth()` using formula = 'y ~ x'
```



In the figure above, we have a scatter plot of the number of completed bookings vs the hour of flight departure. There is a clear trend in the number of completed bookings, with the majority of them being before 3pm. After 3pm, the number of completed bookings decreases as the hour of flight departure becomes later in the day.

Complete Bookings by Customer Preference Combination

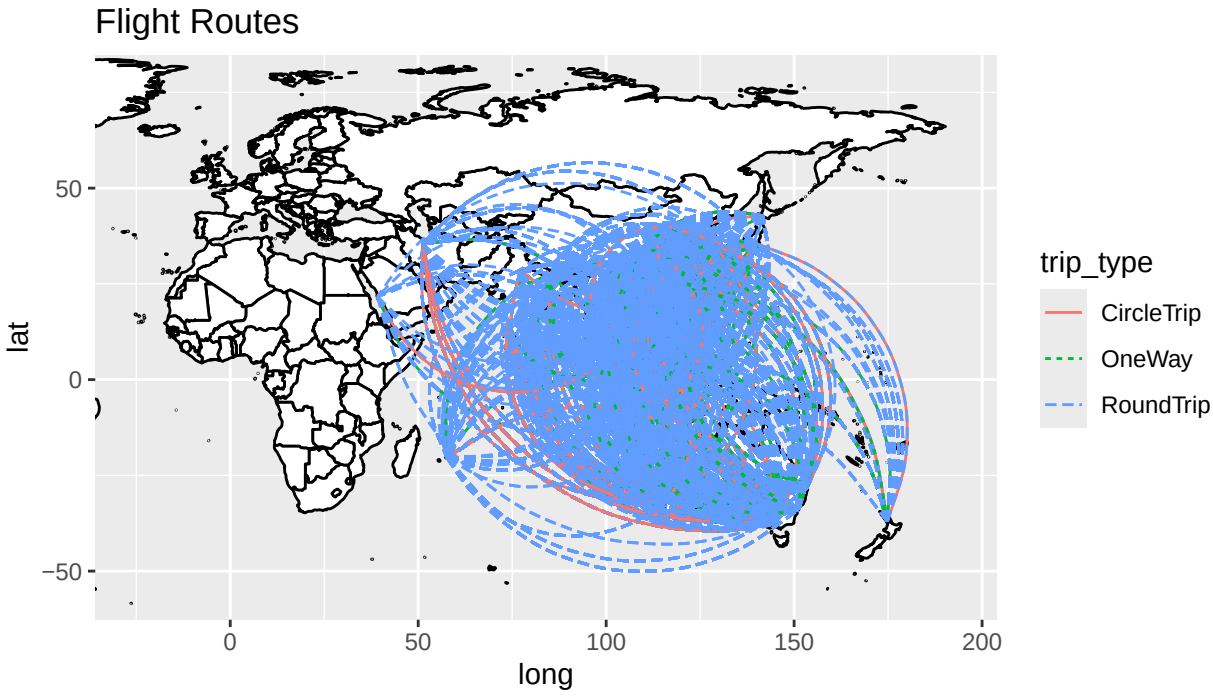


The bar graph above shows the number of complete bookings by customer preference combinations. Having extra baggage only has the highest number of completed bookings, followed by having all three of extra baggage, preferred seats, and in flight meals.

Flight Routes

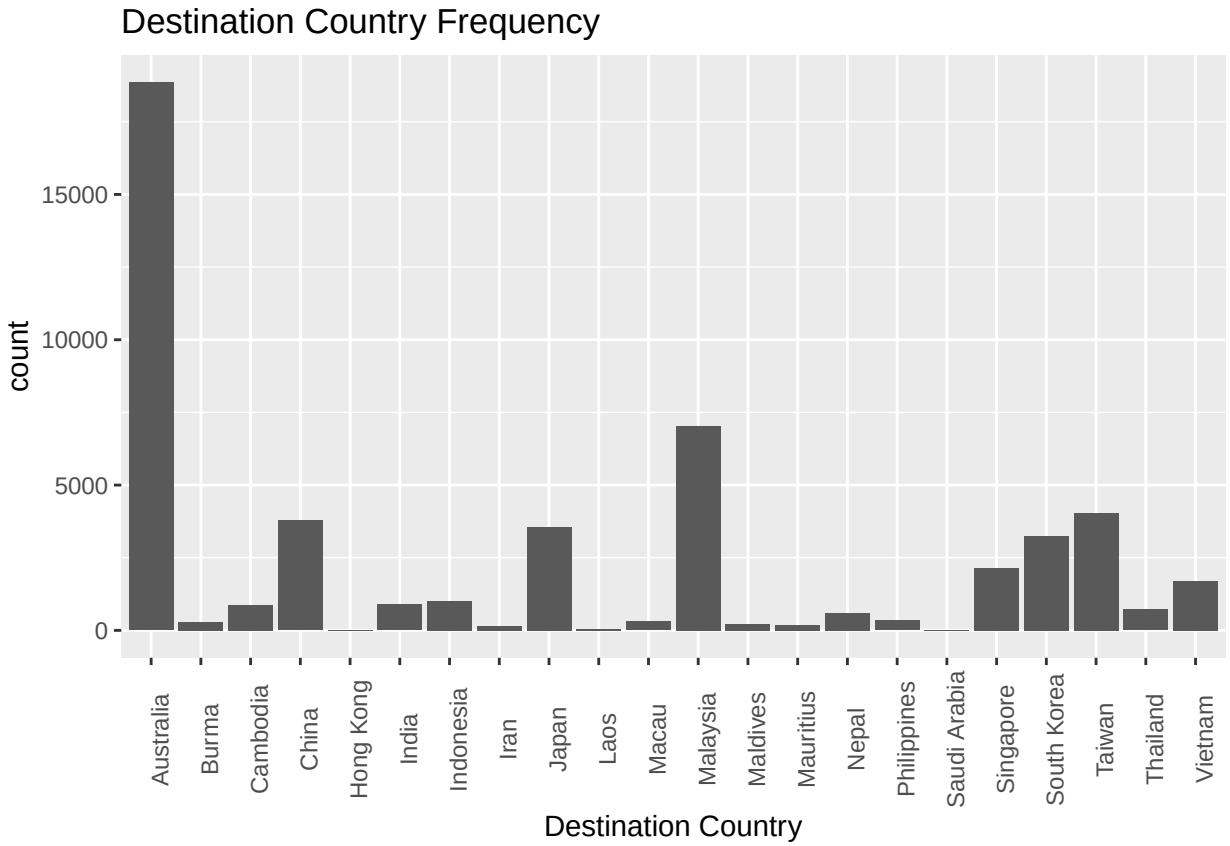
```
route <- data.frame(ogRoute = unique(bookings$route))
routeOrigin <- mutate(route, origin = substr(route$ogRoute, start = 1, stop = 3))
routeDest <- mutate(route, dest = substr(route$ogRoute, start = 4, stop = 6))

mergedOrigin <- merge(x = airports[, c("IATA", "Latitude", "Longitude", "Country")], y = routeOrigin, by
mergedDest <- merge(x = airports[, c("IATA","Latitude", "Longitude", "Country")], y = routeDest, by.x =

colnames(mergedOrigin)[1] <- "origin"
colnames(mergedOrigin)[2] <- "originLat"
colnames(mergedOrigin)[3] <- "originLon"
colnames(mergedOrigin)[4] <- "originCountry"
colnames(mergedDest)[1] <- "dest"
colnames(mergedDest)[2] <- "destLat"
colnames(mergedDest)[3] <- "destLon"
colnames(mergedDest)[4] <- "destCountry"

mergedRoute <- merge(mergedOrigin, mergedDest)
```

```
mergedAll <- merge(mergedRoute[, c("ogRoute", "originLat", "originLon", "destLat", "destLon", "originCou

world_map <- map_data("world")
ggplot(data = world_map) + geom_polygon(aes(x = long, y = lat, group = group), fill = "white", color =
   labs(title = "Flight Routes") +
  coord_quickmap(xlim = c(-25.0, 193.0), ylim = c(-56.0, 78.0))
```



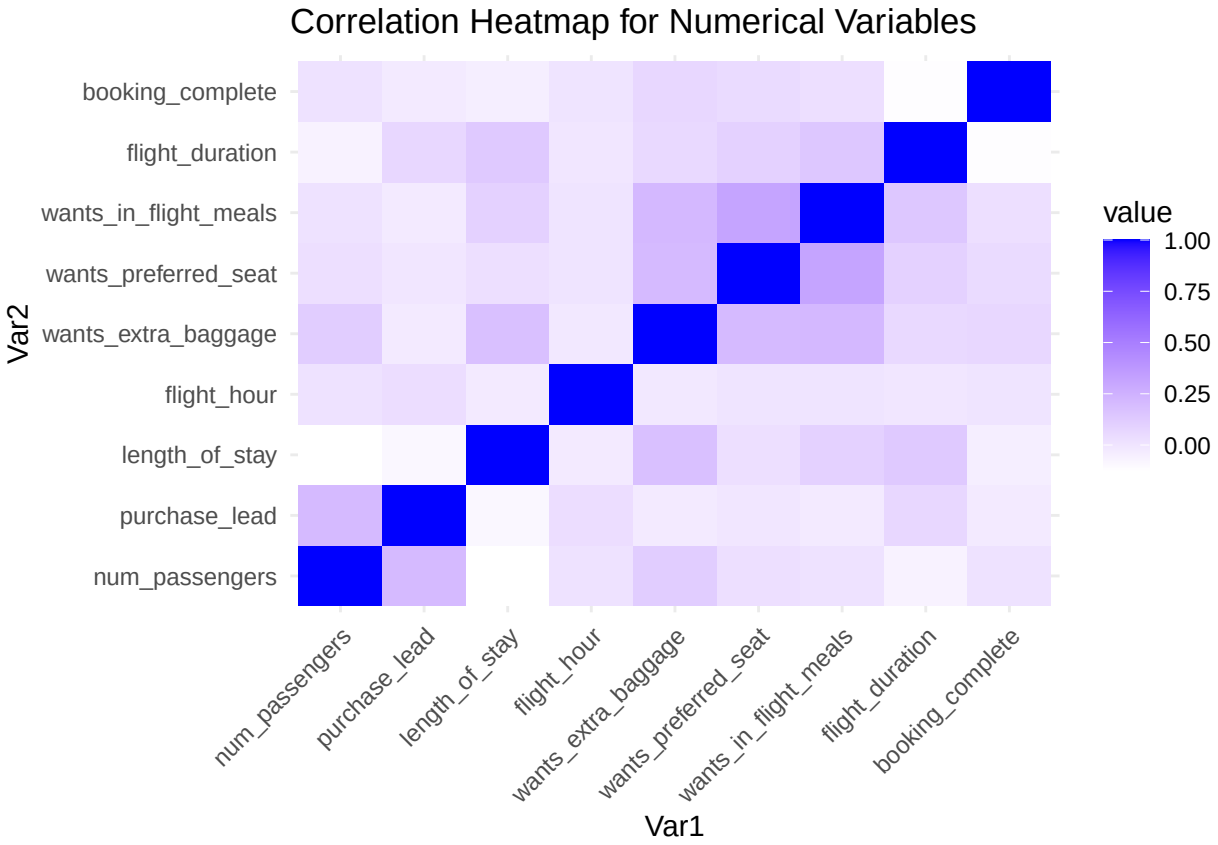
The dataset had the origin and destination airports in an OriginDestination format, so we first had to create two new columns, origin and destination, that only contained either origin or destination. After that, using an airports package called 'airportr', we matched up each origin and destination to its longitude, latitude, and country. Because the dataset was large, we created a new dataframe that only contained the unique origin and destination values, and performed those steps on that new dataframe. Otherwise, the operation would've taken too long to complete. And with that new dataframe, we were able to use the longitude and latitude values of each origin and destination airport in a geom_curve() operation to show the flight routes on a map.

Destination Country Frequency

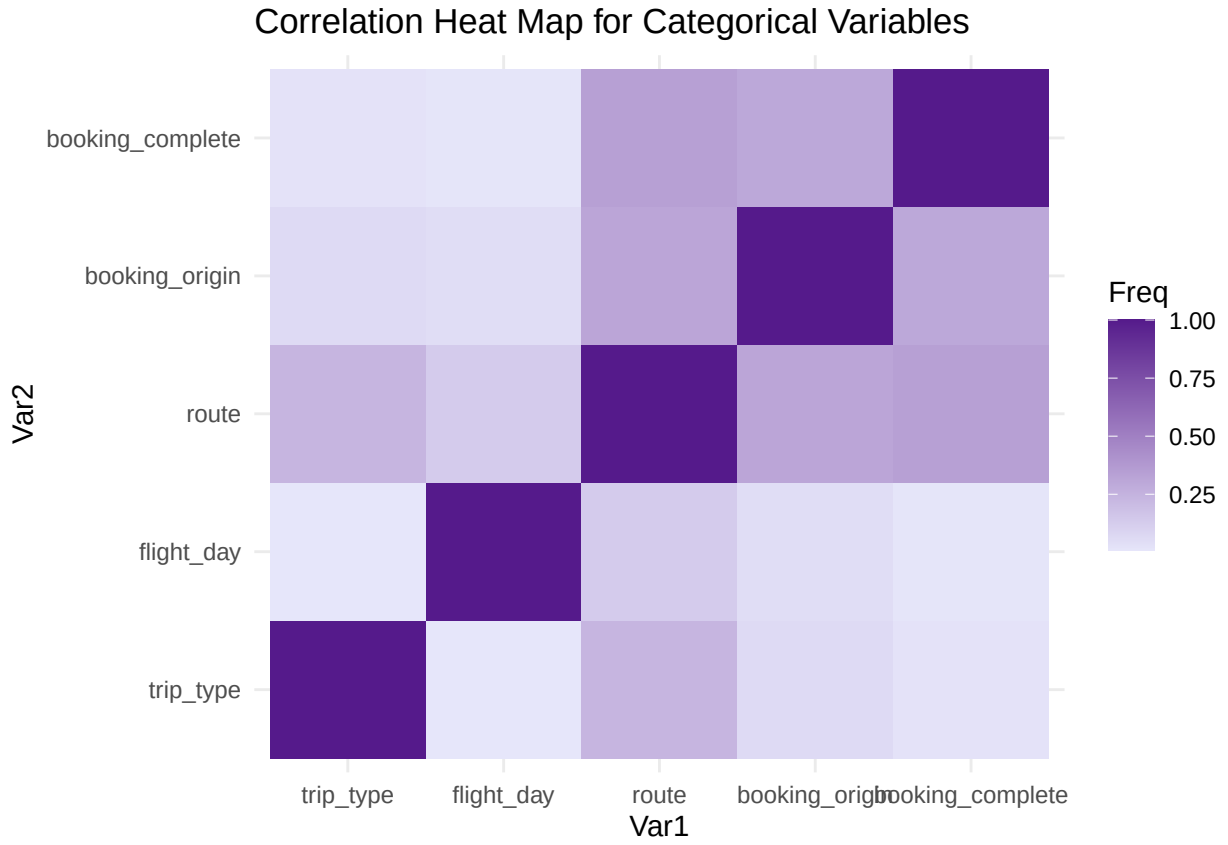


We also then merged the new dataframe with the original dataset by matching them up through the OriginDestination routes, and used the destination countries in a histogram to show the most popular destination countries from the dataset.

Correlation Heat Map of Numerical Variables



Correlation Heat Map of Categorical Variables



Based on the output of the two heatmaps above for correlation between numerical and categorical variables, there is no strong correlation between the variables and booking_complete. However, the heatmap for categorical variables illustrates a slight correlation between booking_complete and route. Due to this revelation, we will be observing this variable to see if there is a true relationship with booking_complete. We also noticed that some variables have a very small correlation with each other. For instance, the heatmap for numerical variables shows a slight correlation between wants_in_flight_meals and wants_perferred seat.

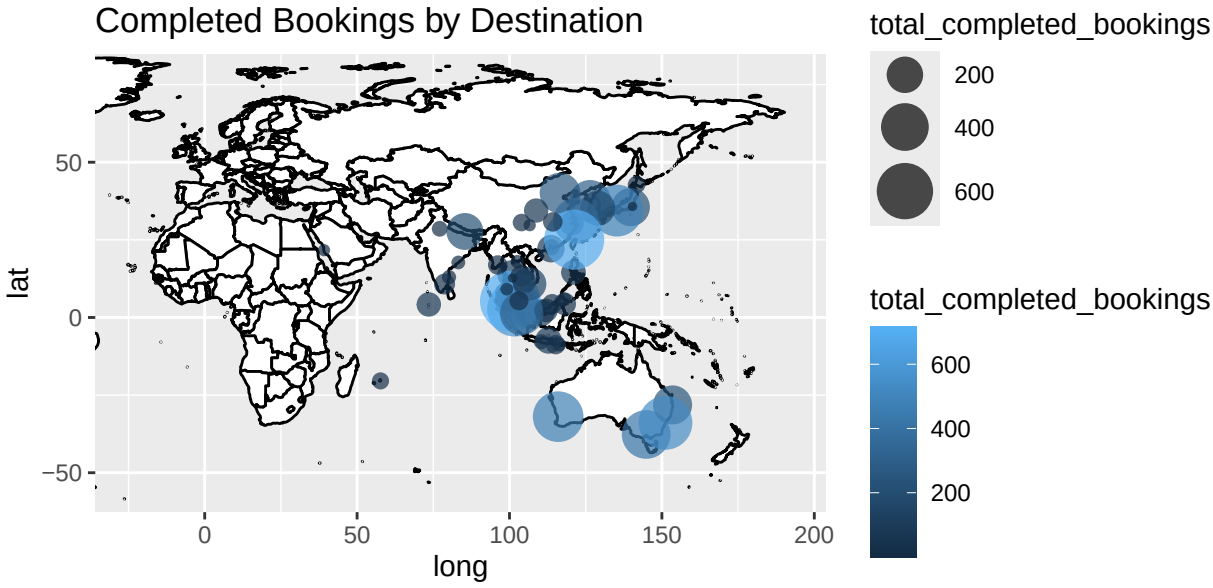
Completed Bookings By Destination

```
# get complete bookings by destination
completed_bookings_by_dest <- bookings |>
  # create new row for destination
  mutate(dest = substr(route, nchar(route) - 2, nchar(route))) |>
  filter(booking_complete == 1) |>
  group_by(dest) |>
  summarize(total_completed_bookings = n())

# get airport subset of IATA, lat, long, and country
airports_subset <- airports %>%
  select(IATA, Latitude, Longitude, Country)

# merge the completed_bookings_by_dest with airports_subset
```

```r
merged_data <- completed_bookings_by_dest %>%
  left_join(airports_subset, by = c("dest" = "IATA"))

# check for any missing values and remove them
merged_data <- merged_data %>%
  filter(!is.na(Latitude) & !is.na(Longitude))

head(merged_data)
```

```
## # A tibble: 6 x 5
##   dest  total_completed_bookings Latitude Longitude Country
##   <chr>                    <int>    <dbl>     <dbl> <chr>
## 1 CKG                          3    29.7      107.  China
## 2 CTS                         19    42.8      142.  Japan
## 3 CTU                         19    30.6      104.  China
## 4 DEL                         12    28.6       77.1 India
## 5 DMK                         95    13.9      101.  Thailand
## 6 DPS                         28    -8.75     115.  Indonesia
```

```r
# get world map
world_map <- map_data("world")

# plot the world map
ggplot(data = world_map) +
  geom_polygon(aes(x = long, y = lat, group = group), fill = "white", color = "black") +
  # plot points for booking destinations, setting size and color to total completed bookings
  geom_point(data = merged_data, aes(x = Longitude, y = Latitude, size = total_completed_bookings, colo
  scale_size_continuous(range = c(1, 10)) +
  # add title
  ggtitle("Completed Bookings by Destination") +
  # set limits for better visualization
  coord_quickmap(xlim = c(-25.0, 193.0), ylim = c(-56.0, 78.0))
```



The map above we have plotted points with color and size based on the total number of completed bookings for a given destination. The larger the size and the lighter the color of the point, the more total bookings there are. We can see in the map that there are more completed bookings in Australia and Southeast Asia.

Classification Models

Since the data set is very large and is made up of a majority of "no" instances for booking_complete, we decided to use under sampling so that the classification models can be trained on a more balanced data set. We counted the number of completed bookings in the data, and sampled an equal number of instances where the booking was not completed.

Under Sampled Data

```
# set the seed
set.seed(167)

# count number of yes and no for booking complete
num_yes <- sum(bookings$booking_complete == 1)
num_no <- sum(bookings$booking_complete == 0)

# define the number of samples to be taken from each class
n_samples <- num_yes  # Adjust this number based on your dataset

# sample indices for "yes" bookings
```

```r
yes_indices <- sample(which(bookings$booking_complete == 1), n_samples)

# sample indices for "no" bookings
no_indices <- sample(which(bookings$booking_complete == 0), n_samples)

# combine the sampled indices
sampled_indices <- c(yes_indices, no_indices)

# create the sampled data set
undersampled_data <- bookings[sampled_indices, ]
```

Naive Bayes

```r
# mutate the data to change the categorical variables to factors
new_bookings <- undersampled_data |>
  mutate(
  sales_channel = as.factor(sales_channel),
  trip_type = as.factor(trip_type),
  flight_day = as.factor(flight_day),
  route = as.factor(route),
  booking_origin = as.factor(booking_origin)
)

# set the seed
set.seed(167)

# split the data into training and test sets
train_indices <- sample(seq_len(nrow(new_bookings)), size = 0.7 * nrow(new_bookings))
train_data <- new_bookings[train_indices, ]
test_data <- new_bookings[-train_indices, ]

# train the naive bayes model
model <- naiveBayes(booking_complete ~ ., data = train_data)

# predict on the test set
predictions <- predict(model, test_data)
```

Naive Bayes Model Evaluation

```
##
## predictions    0    1
##           0 1476  581
##           1  745 1685
```

```
## Accuracy: 0.7044796
```

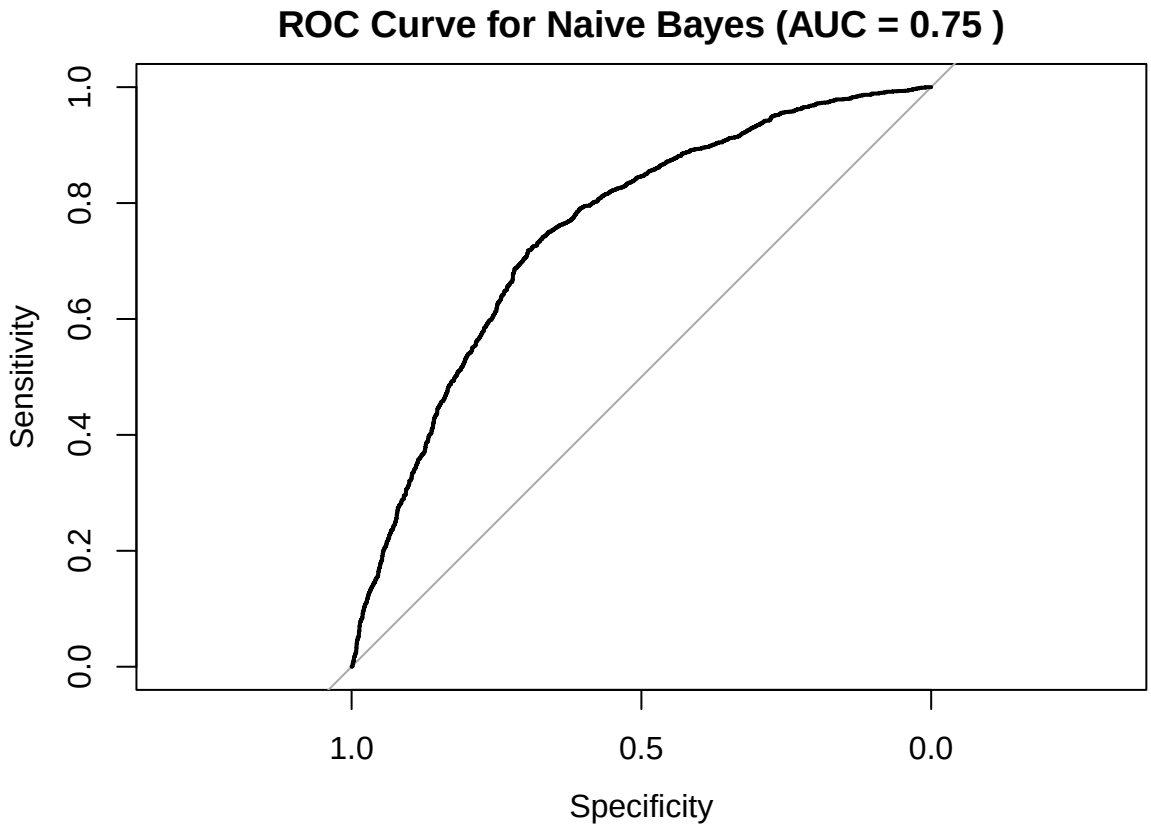
```
## Sensitivity:  0.6934156
```

```
## Specificity:  0.7175498
```

Sensitivity: The proportion of of actual positive cases (booking completions) that are correctly identified by our model. It measures the model's ability to correctly detect TP. In our case we had a sensitivity of 35.98% indicating that the model correctly identifies roughly36% of actual booking completions. This then suggests that the model misses a significant number of actual completions. Specificity: The proportion of actual negative cases (non-booking completions) that are correctly identified by our model. It measures the model's ability to correct detect TN. In our case we had a specificity of 87.85% indicating that the model correctly identifies about 88% of actual non-booking completions. This suggests that the model is quite good at recognizing when a booking is not completed.

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



We created an ROC curve to evaluate the classifier's performance. The area under the curve is about 75%, which means that the model has a 75% chance of correctly distinguishing between non-completed bookings and completed bookings. Overall, we can say that the model can distinguish between positive and negative instances pretty well.

Random Forest

```
# convert relevant columns to factors
rf_df <- undersampled_data |>
  mutate(
    sales_channel = as.factor(sales_channel),
```

```
    trip_type = as.factor(trip_type),
    route = as.factor(route),
    booking_origin = as.factor(booking_origin),
    wants_extra_baggage = as.factor(wants_extra_baggage),
    wants_preferred_seat = as.factor(wants_preferred_seat),
    wants_in_flight_meals = as.factor(wants_in_flight_meals),
    booking_complete = as.factor(booking_complete)
  )

# split data into training and testing sets
set.seed(167)
n <- nrow(rf_df)

# randomly sample indices for the training set (70%)
train.idx <- sample(n, size = n * 0.7)

# create the training and test sets
train <- rf_df[train.idx, ]
test <- rf_df[-train.idx, ]

rf_model <- randomForest(booking_complete ~ num_passengers + sales_channel + trip_type +
                         purchase_lead + length_of_stay + flight_hour + flight_day +
                         wants_extra_baggage + wants_preferred_seat +
                         wants_in_flight_meals + flight_duration,
                         data = train, ntree = 500, mtry = 3, importance = TRUE)

# Make predictions on the test data
rf_predicted_classes <- predict(rf_model, newdata = test)
```

```
# # Calculate misclassification rate
# misclassification_rate <- mean(rf_predicted_classes != test$booking_complete)
# # print(paste("Misclassification rate:", misclassification_rate))
# cat("Misclassification rate:", misclassification_rate, "\n\n")

# Create a confusion matrix
confusion_matrix <- table(Actual = test$booking_complete, Predicted = rf_predicted_classes)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```
print(confusion_matrix)
```

```
##       Predicted
## Actual    0    1
##      0 1408  813
##      1  848 1418
```

```
# Calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.6298195
```

```r
sensitivity <- confusion_matrix[2, 2] / sum(confusion_matrix[2, ])
specificity <- confusion_matrix[1, 1] / sum(confusion_matrix[1, ])

cat("Sensitivity:", sensitivity, "\n")
```

```
## Sensitivity: 0.6257723
```

```r
cat("Specificity:", specificity, "\n\n")
```

```
## Specificity: 0.6339487
```

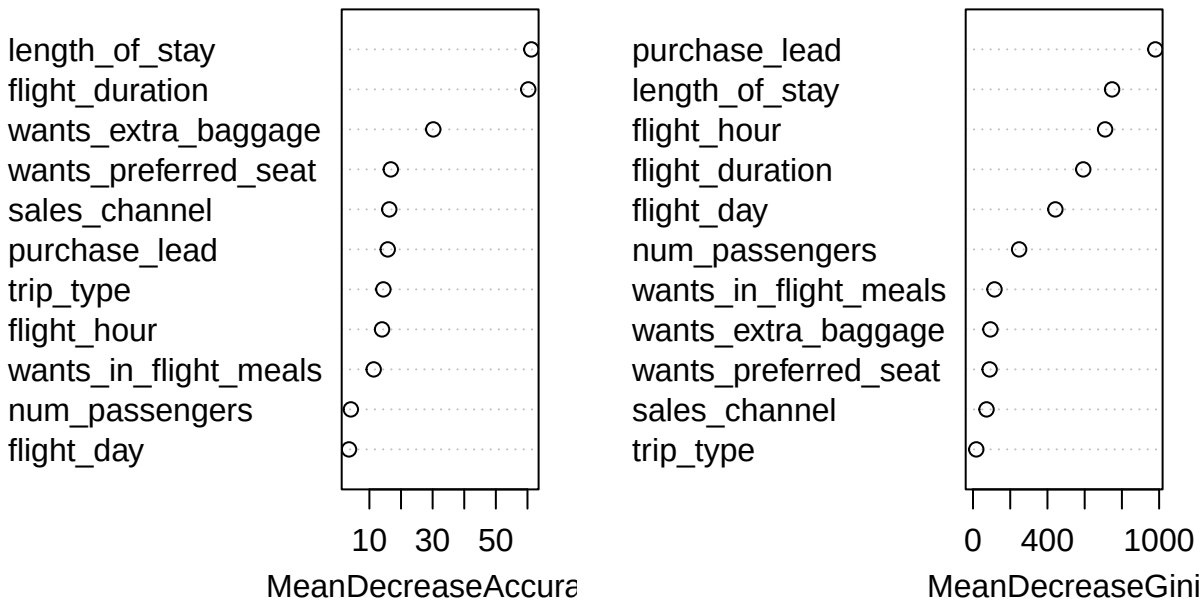
```r
# Print variable importance
print("Variable Importance:")
```

```
## [1] "Variable Importance:"
```

```r
print(importance(rf_model))
```

```
##                              0          1 MeanDecreaseAccuracy
## num_passengers         8.182851 -2.4542652              4.127511
## sales_channel         11.827653 11.7741322             16.284195
## trip_type             11.346166  9.4734171             14.413007
## purchase_lead          3.562661 18.1028692             15.796666
## length_of_stay        48.506296 31.3109433             61.214642
## flight_hour           10.663490  8.8301048             13.995216
## flight_day             1.102044  3.9674922              3.542005
## wants_extra_baggage   25.330783 10.4206631             30.201490
## wants_preferred_seat   8.440044 11.8718192             16.766597
## wants_in_flight_meals 14.167206 -0.1470028             11.356192
## flight_duration       43.988897 37.9734643             60.252236
##                       MeanDecreaseGini
## num_passengers               247.88900
## sales_channel                 72.75870
## trip_type                     17.66507
## purchase_lead                979.50482
## length_of_stay               747.30687
## flight_hour                  709.84309
## flight_day                   442.65673
## wants_extra_baggage           93.56432
## wants_preferred_seat          89.39714
## wants_in_flight_meals        115.54303
## flight_duration              592.37932
```

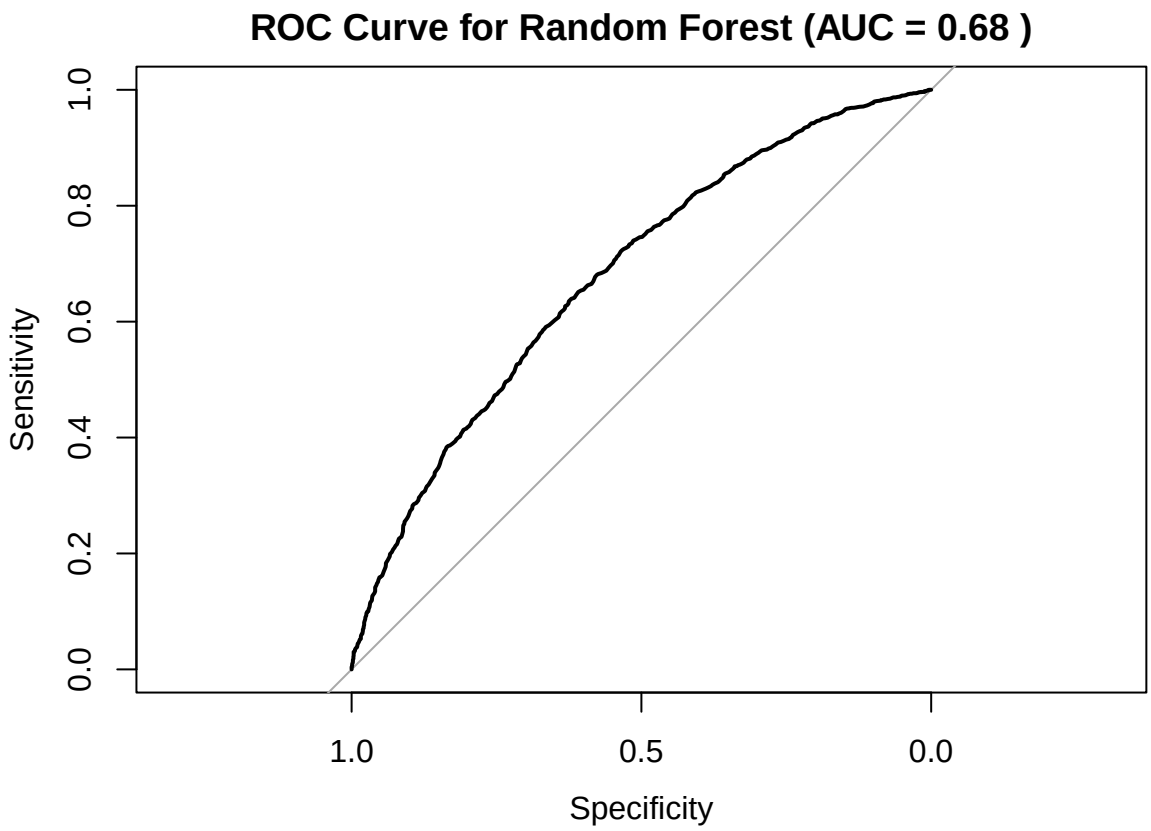
```r
# Plot variable importance
varImpPlot(rf_model)
```



We chose a Random Forest model for our second classification model so that we can see what features are most important for predicting a completed booking. Since it is a non-parametric technique, we don't have to worry about satisfying assumptions for normality and independence. From the confusion matrix we can see that the model has a sensitivity of 62% and specificity of 63%, with an overall accuracy of 63%. Despite the low sensitivity and specificity, we were still able to gain some insight into what variables were most important for deciding the outcome. In the variable importance plot, we can see that length_of_stay is the most important in maintaining accuracy because it has the highest mean decrease accuracy measurement. In the plot for Mean Decrease Gini, we can see that purchase_lead is the most important in splitting the data to classify whether it is a yes or no. When we consider both metrics of Mean Decrease Accuracy and Mean Decrease Gini, length_of_stay, flight_duration, purchase_lead, flight_hour, and wants_extra_baggage seem to be the top five most important variables.

```
## Setting direction: controls > cases
```

```
## AUC: 0.6792028
```



For a Random Forest model, the ROC curve helps in understanding how well the model performs across different thresholds, and the AUC provides a single metric that allows us to evaluate its overall performance. Since the Area Under the Curve (AUC) is 0.68, we can conclude that the model has some ability to distinguish between positive and negative instances. However, it is overall not a very good model.

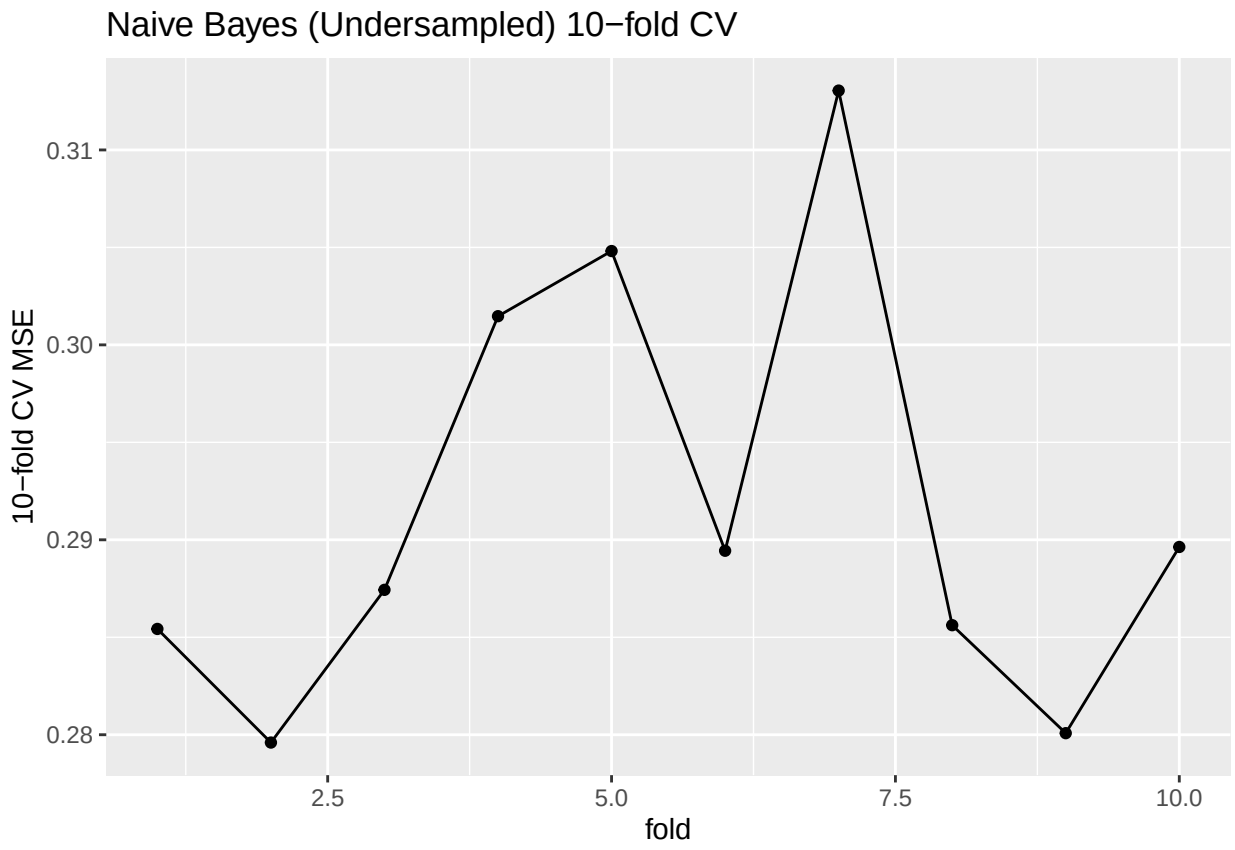
Model Evaluation

In addition to classification evaluation metrics, we have also performed 10 Fold Cross Validation to evaluate our models.

Naive Bayes 10 Fold Cross Validation

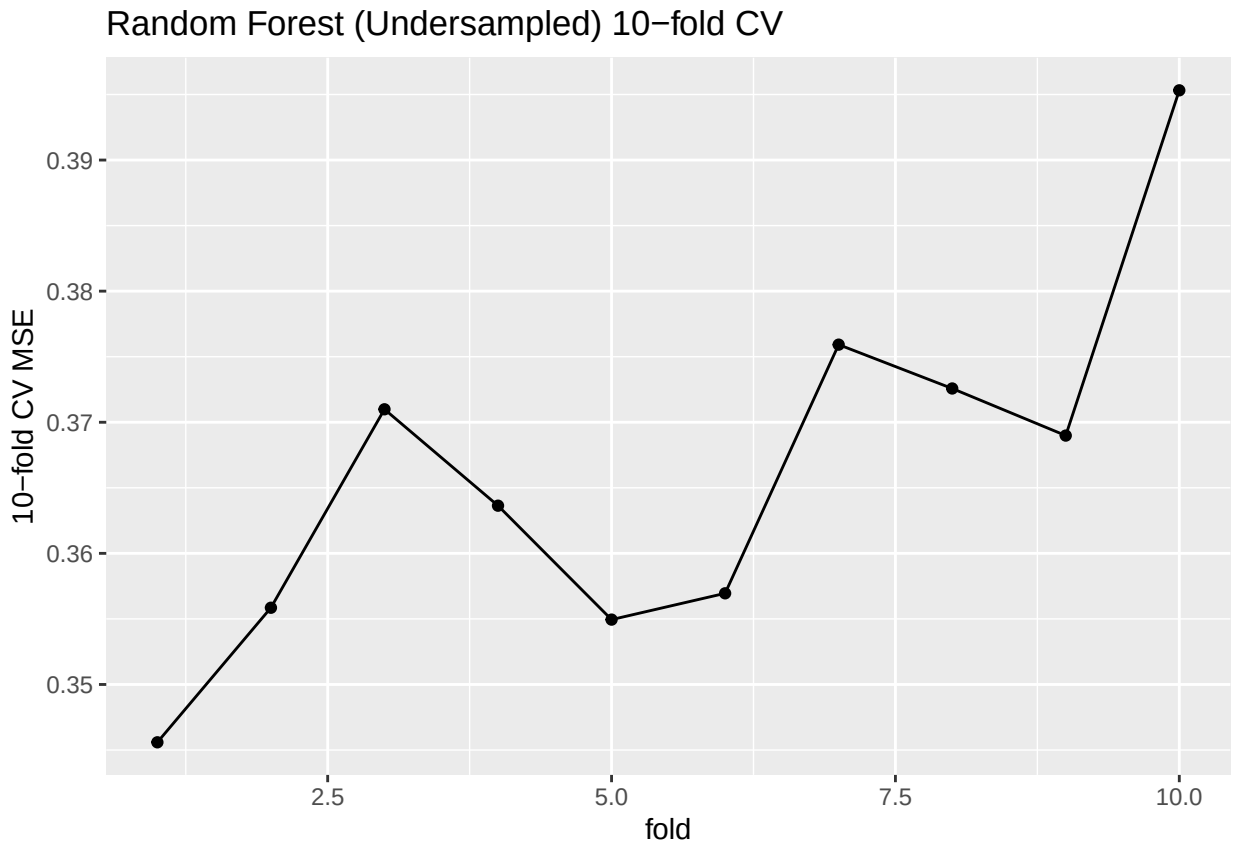
```
# set seed
set.seed(167)

new_bookings$booking_complete <- as.factor(new_bookings$booking_complete)

# Function for manual 10-fold cross-validation to calculate MSE
cross_validate_naive_bayes_mse <- function(data, k) {
  folds <- createFolds(data$booking_complete, k = k)
  mses <- numeric(k)

  for (i in 1:k) {
    test_indices <- folds[[i]]
```

```r
    train_data <- data[-test_indices, ]
    test_data <- data[test_indices, ]

    model <- naiveBayes(booking_complete ~ ., data = train_data)
    predictions <- predict(model, test_data, type = "raw")  # Get probabilities
    # Convert probabilities to numeric predictions (0 or 1)
    numeric_predictions <- as.numeric(predictions[,2] > 0.5)
    numeric_actual <- as.numeric(test_data$booking_complete) - 1  # Convert factor to numeric

    mses[i] <- mean((numeric_predictions - numeric_actual)^2)
  }

  return(mses)
}

# Run the cross-validation
cv_mse <- cross_validate_naive_bayes_mse(new_bookings, k = 10)
cv.df <- tibble(fold = 1:10, MSE.cv = cv_mse)

ggplot(data = cv.df, mapping = aes(x = fold, y = MSE.cv)) +
geom_point() + geom_line() + ylab("10-fold CV MSE") +
  labs(title = "Naive Bayes (Undersampled) 10-fold CV")
```



Random Forest 10 Fold Cross Validation

```
# Function for manual 10-fold cross-validation to calculate MSE on random forest model
cross_validate_random_forest_mse <- function(data, k) {
  folds <- createFolds(data$booking_complete, k = k)
  mses <- numeric(k)

  for (i in 1:k) {
    test_indices <- folds[[i]]
    train_data <- data[-test_indices, ]
    test_data <- data[test_indices, ]

    model <- randomForest(booking_complete ~ num_passengers + sales_channel + trip_type +
                          purchase_lead + length_of_stay + flight_hour + flight_day +
                          wants_extra_baggage + wants_preferred_seat +
                          wants_in_flight_meals + flight_duration,
                          data = train_data, ntree = 500, mtry = 3, importance = TRUE)
    # get predictions
    predictions <- predict(model, test_data)  # Get class predictions

    # convert factor to numeric
    numeric_predictions <- as.numeric(predictions) - 1
    numeric_actual <- as.numeric(test_data$booking_complete) - 1

    mses[i] <- mean((numeric_predictions - numeric_actual)^2)
  }

  return(mses)
}

# Run the cross-validation
set.seed(167)  # For reproducibility
cv_mse <- cross_validate_random_forest_mse(rf_df, k = 10)
cv.df <- tibble(fold = 1:10, MSE.cv = cv_mse)

ggplot(data = cv.df, mapping = aes(x = fold, y = MSE.cv)) +
geom_point() + geom_line() + ylab("10-fold CV MSE") +
    labs(title = "Random Forest (Undersampled) 10-fold CV")
```



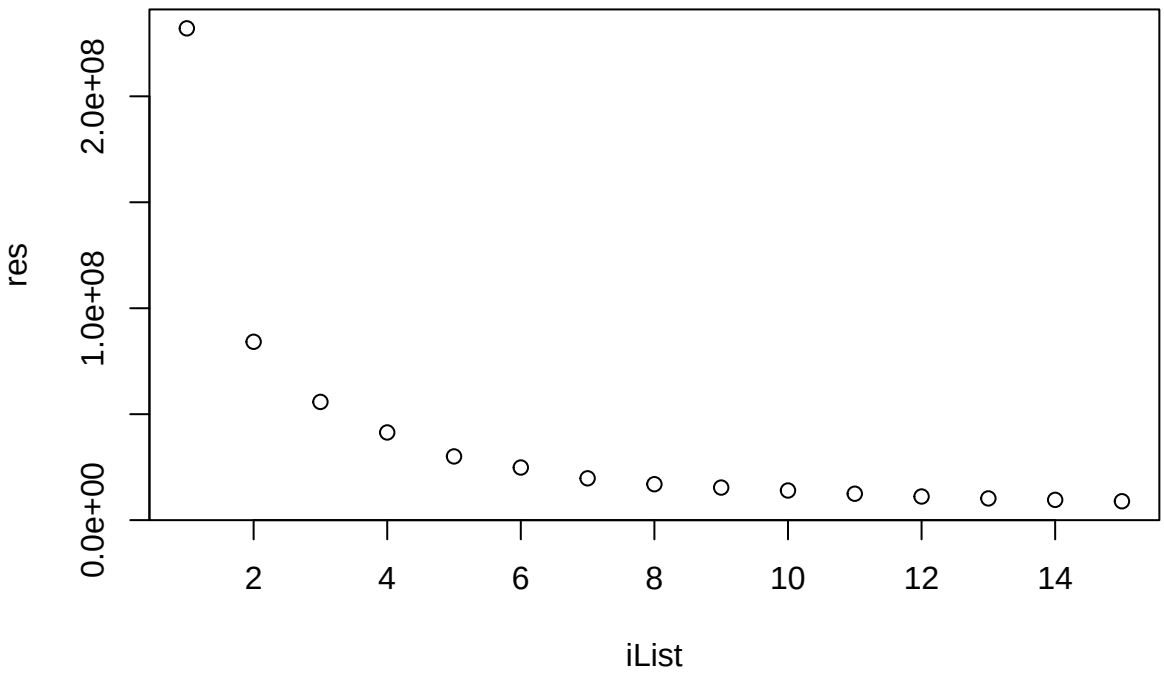
For the Naive Bayes, the 10-fold cross validation ended up increasing the average MSE per fold, however, it reduced the problem prior to undersampling where the last 2 folds tended to have much higher MSEs due to data with low instances of bookings completed. Furthermore, comparing the two under sampled models directly we were able to see that the Naive Bayes had lower MSEs than the random forest model leading us to believe that it was the better model in terms of predicting whether bookings would be completed or not. This cross-validation also showed that there were certain areas of the data that definitely performed better which we believe is once again due to the fact that there weren't many instances of bookings being completed in the data set as a whole.

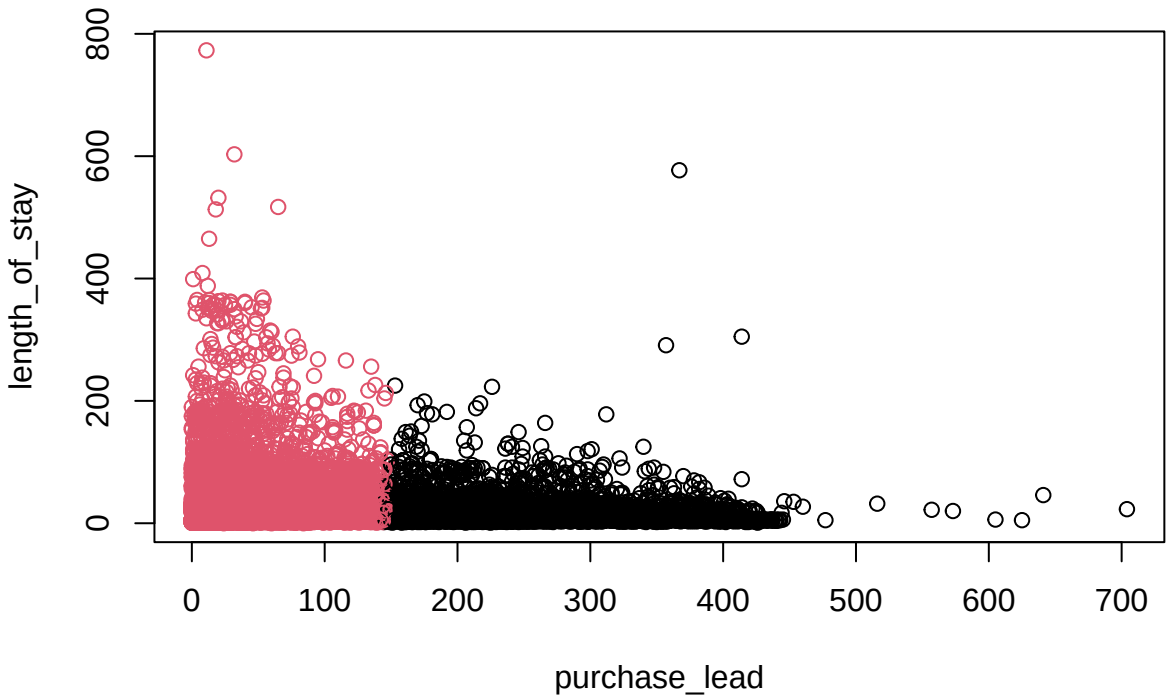
Limitations

Model Attempts

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 1250000)
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 1250000)
```

```
## Warning: did not converge in 10 iterations
## Warning: did not converge in 10 iterations
## Warning: did not converge in 10 iterations
## Warning: did not converge in 10 iterations
```


res
0.0e+00
1.0e+08
2.0e+08
2
4
6
8
10
12
14
iList



The dataset had the origin and destination airports in an OriginDestination format, so we first had to create two new columns, origin, and destination, that only contained either origin or destination. After that, using an airports package called 'airportr', we matched each origin and destination to its longitude, latitude, and country. Because the dataset was large, we created a new data frame that only contained the unique origin and destination values and performed those steps on that new data frame. Otherwise, the operation would've taken too long to complete. With that new data frame, we were able to use the longitude and latitude values of each origin and destination airport in a geom_curve() operation to show the flight routes on a map. We then merged the new data frame with the original dataset by matching them up through the OriginDestination routes and used the destination countries in a histogram to show the most popular destination countries from the dataset.

Conclusion

In conclusion, we were able to achieve our project goal of determining what factors influence an airline booking. Although the exploratory data analysis showed some trends in our data, the correlation heat maps showed that there was no significant correlation between the variables and an airline booking, as well as no correlation between the variables themselves. Before creating our classification models, we had to undersample the data to create a more balanced training and test set with equal "yes" and "no" instances for booking_complete. For our models, Naive Bayes with under-sampling had a relatively high sensitivity of 0.69 and a relatively low specificity of 0.72. Random Forest with under-sampling had a sensitivity of 0.62 and a specificity of 0.63. Although both models did not perform very well, when we compared the two directly using classification metrics and the 10-fold cross-validation, we found that the Naive Bayes model proved to be better. However, based on the variable importance metrics from the Random Forest Model, we were also able to determine that passengers value the following factors the most: length_of_stay flight_duration, purchase_lead, flight_hour, and wants_extra_baggage. There were some limitations to our project due to

the fact that there are many fewer instances of people completing a booking than not completing a booking in our data set. Despite this, we were still able to identify factors and trends that influenced people to complete bookings.

Authors Contributions

Project Intro/Description

- Brian

EDA

- Lindsay
  - Number of Passengers vs. Purchase Lead Time
  - Bookings By Flight Hour, Sales Channel, and Booking Status
  - Correlation Heat Map of Categorical Variables
- Amy
  - Bookings vs. Hour of Flight Departure
  - Complete Bookings by Customer Preference Combination
  - Completed Bookings By Destination
- Emlyn
  - Flight Routes
  - Destination Country Frequency
- Adelric
  - Correlation Heat Map of Numerical Variables

Models

- Lindsay (Naive Bayes)
- Amy (Random Forest)

Classification Evaluation for Models

- Lindsay (Naive Bayes confusion matrix)
- Amy (Random Forest confusion matrix)
- Brian (sensitivity, specificity, ROC curve)

Model Evaluation with 10 fold CV

- Adelric
  - Naive Bayes
  - Random Forest

Limitations

- Emlyn

Conclusion

- Adelric

Data/Code Availability

Link to Kaggle dataset: Airlines Booking Link to Google Drive: [STAT167 Group 11 Final Project Google Drive] (https://drive.google.com/drive/folders/1fI4_bw7bWNu-Ppzi-fGRair9oh38WO13?usp=sharing)