George Mason University

Flight Delays

Final Project

Mitchell Breeden
STAT-515-009
Dr. Niloofar Ramezani
December 7, 2022

Introduction

When my travel includes flying, I always know there is a possibility my flight will be delayed. On a recent trip, after yet another delay, I asked myself what factors influence these seemingly innocuous delays? Was it the date, day of week, airline, location, or even airport? This inspired me to utilize my data analytic skills to find an answer. I was able to find a dataset on Kaggle entitled "2015 Flight Delays and Cancellations" which showed data from the U.S. Department of Transportation's Bureau of Transportation Statistics on domestic flights by large air carriers in January of 2015. This dataset contained 469,968 rows and 31 features encompassing every aspect of a flight. For this project, I chose to use the variables identified above as an average airline passenger would likely think of the same ones: date, day of week, airline, location, and airport. Each of these variables will be discussed in greater detail, with the goal being to turn this data into consumable formats for the traveling public.

Data Cleaning

Before beginning my analysis, there were some necessary data cleaning steps that needed to be completed. After importing the flight data, there were two additional files that needed to be imported: airline and airport mappings. These files mapped the airline/airport IATA (International Air Transport Association) codes to a more readable version (e.g., "OO" = "Skywest Airlines Inc."). Once all necessary files were imported and merged, I was able to start cleaning specific columns. This involved first dropping all flights with no departure time as this meant I could not determine if there was a delay (leaving 458,311 rows). This also delivered a total number of delays of 176,627 which is 38.54% of flights tracked. Next, the month/day of week was transformed from numeric to a string (e.g., "1" = "Monday"). The FAA requires delays on the tarmac to be less than three hours (longer delays and passengers must be afforded

the opportunity to deplane). With this in mind, the following 7 features were added to assist in the analysis: DELAYED_1 (< 1 Hour Delay), DELAYED_2 (1-2 Hour Delay), DELAYED_3 (2-3 Hour Delay), DELAYED_4 (> 3 Hour Delay), is_DELAYED (0 = On-Time, 1 = Delayed), ON_TIME (0 = Delayed, 1 = On-Time), and region (state abbreviations converted to full state names).

Correlation

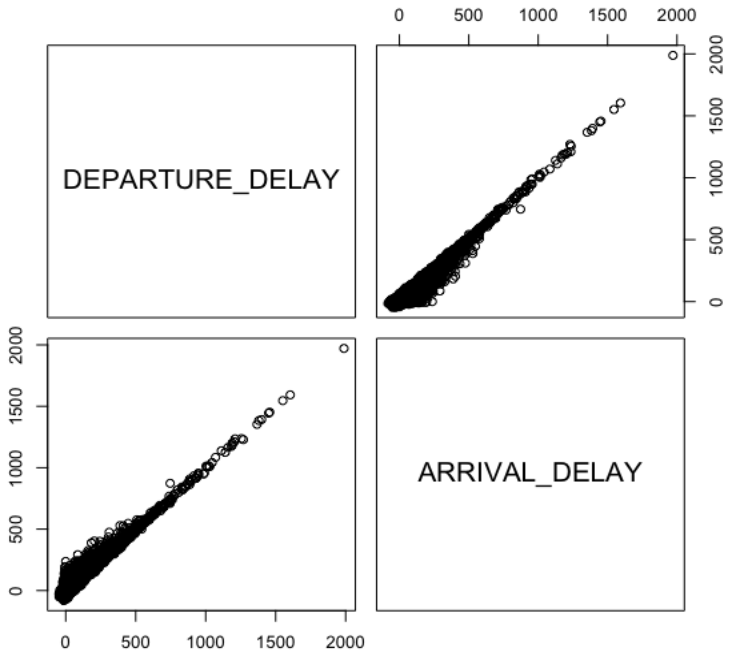

Figure 1. Scatterplot matrix of departure delays vs arrival delays

To start the analysis, the type of delay had to be chosen. Figure 1 displays a scatterplot matrix of the departure delays and the arrival delays appearing to show a very strong correlation. When testing the exact correlation, the value is 0.938116, meaning most flights delayed on departure were also delayed on arrival. Unless there is a connecting flight, most people only care about departure delays; since the correlation is so strong, departure delays were chosen as the target variable.

Date

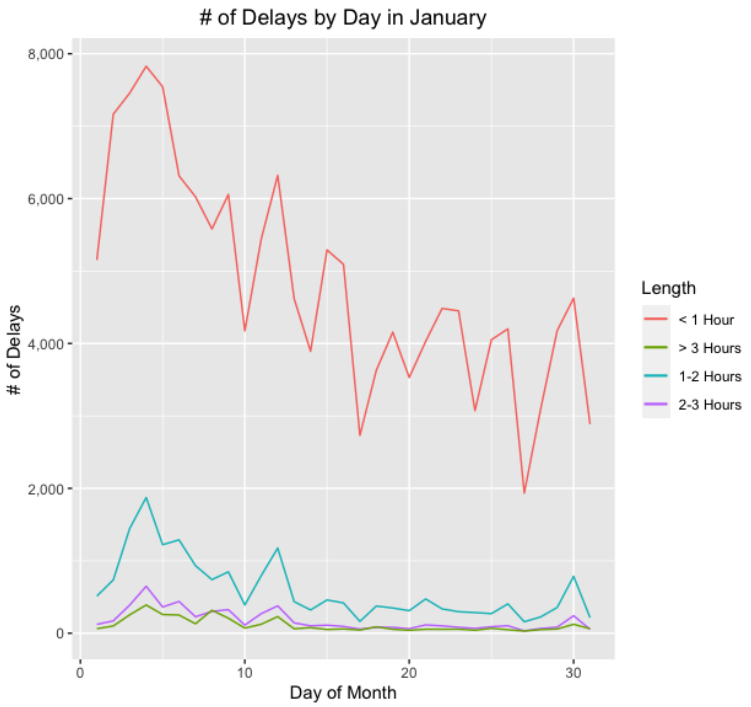

Figure 2. Line graph showing number of delays in January by Length

The first variable analyzed was the date to see if there was a specific day(s) in January with an above average number of delays. As seen in Figure 2, the delays have been plotted on a line graph as the total number of delays in each of the four length groups. There was a noticeable spike in delays at the beginning of January, which was likely attributable to holiday travel. Additionally, even with a large fluctuation in totals, there are significantly more delays in the < 1 hour length group than the other three groups. With a narrowed timeframe for the delays identified, it was time to analyze the days of the week to determine the distribution of the delays.

Day of Week

Next, the day of week is analyzed as a subset of the month of January. In Figure 3, the stacked bar chart demonstrates the number of delays by day of week for the four lengths: Friday

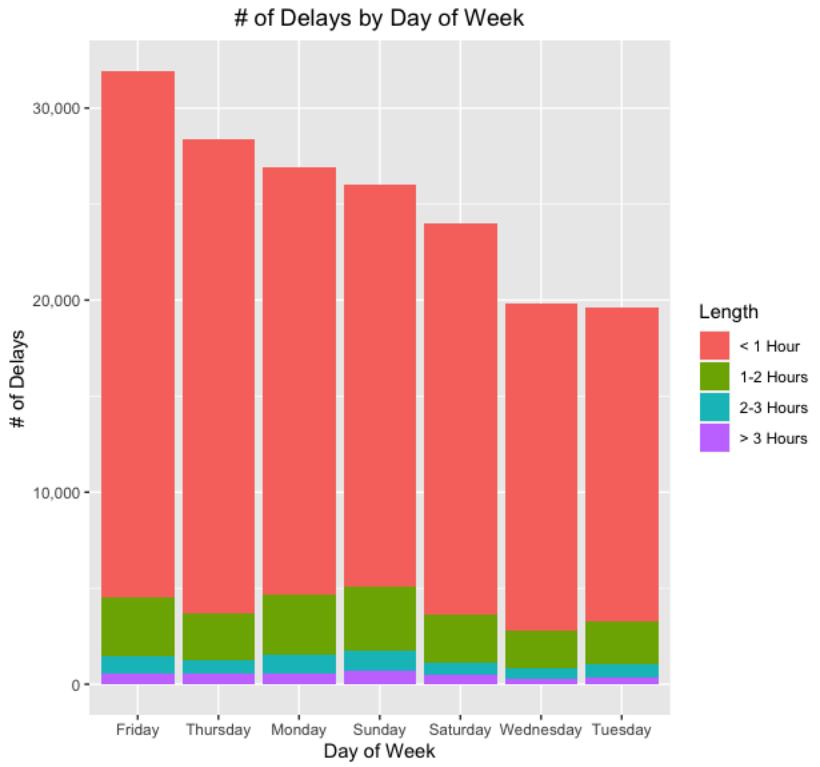

Figure 3. Stacked bar chart showing number of delays by day of week by length

is depicted as having the most delays, while Tuesday has the least number of delays. This finding was not surprising given society's Monday thru Friday work week, and it helps to explain why Sunday has the largest number of greater than 3-hour delays: there are more returning passengers. While the date and day of travel is important, the likelier question for airliner travelers is where their favorite airline falls on the delay list.

Airline

While the 21st century has seen numerous airline mergers, it has also seen countless new airlines begin operations; however, airlines have different routes they service, and different sized aircraft fleets. Due to this variation in the number of flights per airline, Figure 4 had to be plotted by the percent of delays attributed to each length group to avoid bias. By doing this, each airline

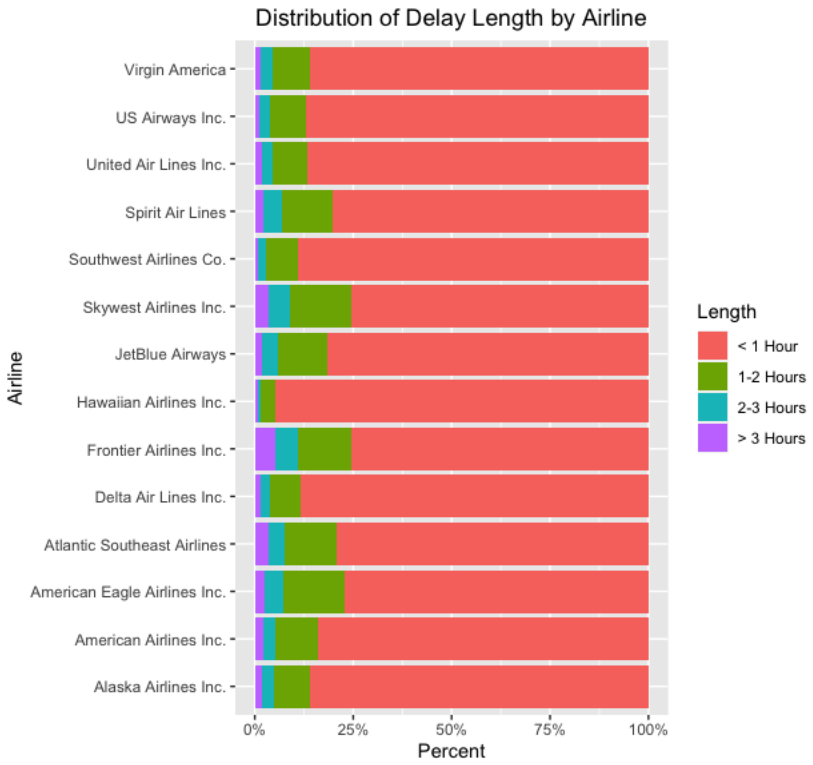

Figure 4. Stacked bar chart showing percent of delays by length by Airline

is equally represented, and it is visible that Frontier Airlines Inc. had the most delays greater than 3 hours. It also shows that all airlines have more than 75% of their flight delays being less than 1 hour. Now that we know which airlines have the most delays, it is important to know where these delays are taking place in the United States.

Location

The flight data covers all 50 states sufficiently enough to be able to plot their delays. Figure 5 is a linked micromap showing the number of delays and on-time flights by each state. In the top of the graph, the five least delayed states are shown as Delaware, Maine, West Virginia, Vermont, and New Hampshire. At the bottom of the graph, the five most delayed states are shown as Georgia, Florida, Illinois, California, and Texas. The on-time graph shows a similar distribution to the delays which is likely caused by the states having more air travel. Knowing

which state is the most delayed is helpful; but, there are sometimes multiple large airports in a single state, so it is vital to know which ones could be influencing the number of delays.

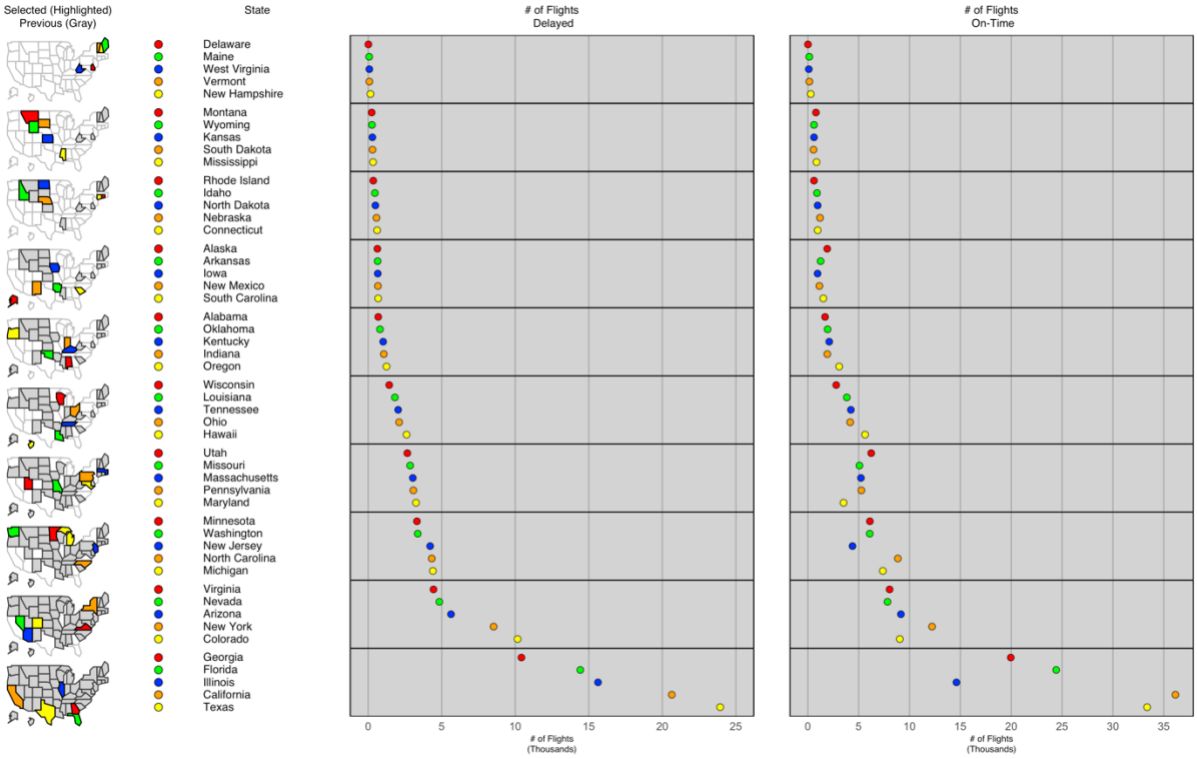

Figure 5. *Linked micromap depicting number of delayed flights and number of on-time flights by state*

Airport

The dataset has a total of 312 airports. This number is quite large and many of these airports are regional which are not significant to most travelers. To alleviate this, Figure 6 shows the top ten delayed airports in order broken down by length group. It is seen that Chicago O'Hare International Airport is the most delayed which also correlates to Illinois being in the top 5 most delayed states in Figure 5. The least delayed airport is seen as Newark Liberty International Airport. This graph helps to give travelers a better understanding of the delays at an airport they may be considering departing from and possibly consider an alternative. With all of this information, it is feasible to see if these data points can be used to accurately predict a flight delay.

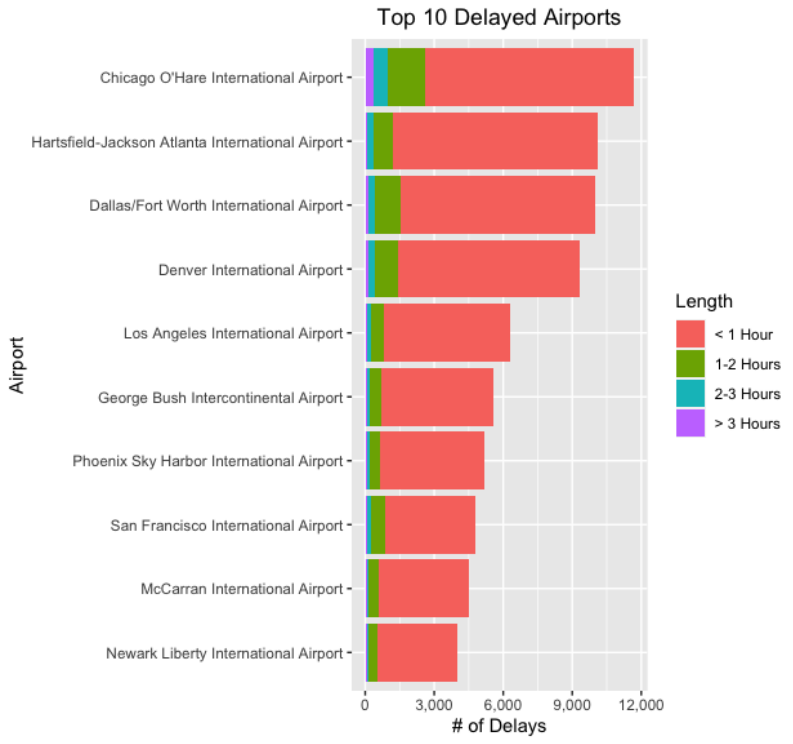

Figure 6. Stacked bar chart showing the number of delays by airport by length

Random Forest

Random forest was chosen as the model type for this problem because of how it handles classification problems. For this case, the predicted variable is either 0 for no delay or 1 for delay. The target variable was assigned to a factored is_DELAYED and the other variables were specified as ORIGIN_STATE, ORIGIN_AIRPORT, AIRLINE, DAY_OF_WEEK, DAY. These are the same variables that were analyzed in the exploratory data analysis. The data frame needed to be a random 5,000 rows to avoid a memory error and to ensure a quality sampling.

After dropping null values, the random forest function was run. The OOB estimate of error rate was 35.34%. When the random forest was used to predict values in the test set it achieved an accuracy of 64.44%. This number is quite low but may be attributed to the variable selection.

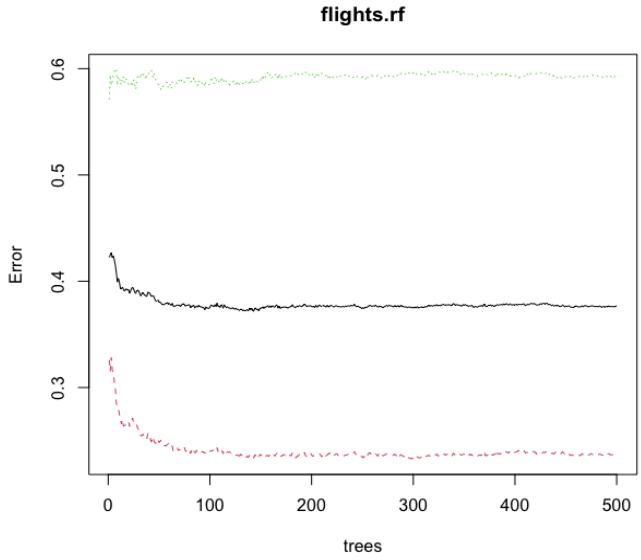

Figure 7. Error chart for random forest model

The variable importance plot as shown in Figure 8 displays the 5 variables in their order of importance to the model. The most important being the DAY and the least important being the DAY_OF_WEEK. This is very helpful in analyzing which variables were of use and which could be dropped in favor of more influential features in future iterations.

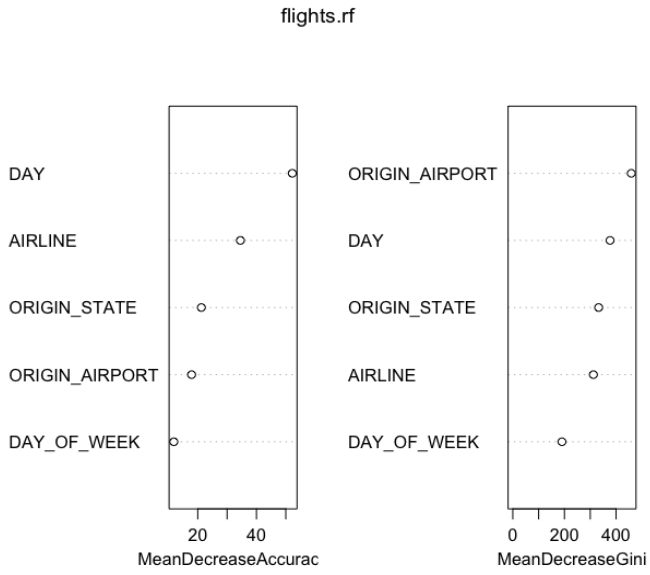

Figure 8. Variable importance plot from random forest model

Conclusion

This dataset contained many useful insights. By examining delays based on date, day of week, airline, location, and airport, inferences can be made about what factors make a delay more or less likely. However, when these factors are inputted into a classification model it becomes clear they are not all equally significant. This is where other variables in the dataset could come into play such as departure time and distance. These added variables along with more computing power could lead to a smaller error rate when running the model. Continued work on this dataset could provide more information as there are multiple ways to look at the data (e.g., percentage of flights by state and airport instead of number). For this analysis, the five variables were chosen based on relevance to passengers and are sufficient in producing a picture of delays.

References

A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Kuhn M (2022). _caret: Classification and Regression Training_. R package version 6.0-93, https://CRAN.R-project.org/package=caret.

Payton, Q., Weber, M., McManus, M., Olsen, T., & Kincaid, T. (2015, August 31). *Linked Micromaps*. The Comprehensive R Archive Network. Retrieved December 7, 2022, from https://cran.r-project.org/web/packages/micromap/vignettes/Introduction_Guide.pdf

Quinn C. Payton, Michael G. McManus, Marc H. Weber, Anthony R. Olsen, and Thomas M. Kincaid (2015). micromap: A Package for Linked Micromaps. Journal of Statistical Software, 63(2), 1-16. URL http://www.jstatsoft.org/v63/i02/.

Random Forest approach in R programming. GeeksforGeeks. (2020, June 5). Retrieved December 7, 2022, from https://www.geeksforgeeks.org/random-forest-approach-in-r-programming/

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

RStudio Team (2022). RStudio: Integrated Development Environment for R. RStudio, PBC, Boston, MA URL http://www.rstudio.com/.

Transportation, D. of. (2017, February 9). *2015 flight delays and cancellations*. Kaggle. Retrieved December 7, 2022, from https://www.kaggle.com/datasets/usdot/flight-delays?select=airlines.csv

Wickham H, François R, Henry L, Müller K (2022). _dplyr: A Grammar of Data Manipulation_. R package version 1.0.10, https://CRAN.R-project.org/package=dplyr.

Wickham H, Girlich M (2022). _tidyr: Tidy Messy Data_. R package version 1.2.1, https://CRAN.R-project.org/package=tidyr.

Wickham H, Seidel D (2022). _scales: Scale Functions for Visualization_. R package version 1.2.1, https://CRAN.R-project.org/package=scales.

Appendix: Source Code

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyr)
library(ggplot2)
library(scales)
library(micromap)

## Loading required package: maptools

## Loading required package: sp

## Checking rgeos availability: TRUE
## Please note that 'maptools' will be retired during 2023,
## plan transition at your earliest convenience;
## some functionality will be moved to 'sp'.

## Loading required package: RColorBrewer

## Loading required package: rgdal

## Please note that rgdal will be retired during 2023,
## plan transition to sf/stars/terra functions using GDAL and PROJ
## at your earliest convenience.
## See https://r-spatial.org/r/2022/04/12/evolution.html and https://github.c
om/r-spatial/evolution
## rgdal: version: 1.6-2, (SVN revision 1183)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.5.3, released 2022/10/21
## Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/4.2-ar
m64/Resources/library/rgdal/gdal
##  GDAL does not use iconv for recoding strings.
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 9.1.0, September 1st, 2022, [PJ_VERSION: 910]
## Path to PROJ shared files: /Library/Frameworks/R.framework/Versions/4.2-ar
m64/Resources/library/rgdal/proj
## PROJ CDN enabled: FALSE
## Linking to sp version:1.5-1
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
```

```
## use options("rgdal_show_exportToProj4_warnings"="none") before loading sp
or rgdal.

## Loading required package: sf

## Linking to GEOS 3.10.2, GDAL 3.4.2, PROJ 8.2.1; sf_use_s2() is TRUE

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:ggplot2':
##
##     margin

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

library(caret)

## Loading required package: lattice

data("USstates")
###CLEANING###

#Import Data
df_flights <- read.csv("flights.csv")
df_airport_origin <-read.csv("airports.csv")
df_airport_destination <- read.csv("airports.csv")
df_airlines <- read.csv("airlines.csv")

#Differentiate Origin and Destination Airports
colnames(df_airport_origin) <- paste0("ORIGIN_", colnames(df_airport_origin))
colnames(df_airport_destination) <- paste0("DESTINATION_", colnames(df_airpor
t_destination))

#Merge Data Frames
df <- left_join(df_flights, df_airport_origin, by = c("ORIGIN_AIRPORT" = "ORI
GIN_IATA_CODE"))
df <- left_join(df, df_airport_destination, by = c("DESTINATION_AIRPORT" = "D
ESTINATION_IATA_CODE"))
df <- left_join(df, df_airlines, by = c("AIRLINE" = "IATA_CODE"))

#Drop Duplicate Columns and Rename
df <- select(df, -AIRLINE, -ORIGIN_AIRPORT, -DESTINATION_AIRPORT)
colnames(df)[colnames(df) == "ORIGIN_AIRPORT.y"] = "ORIGIN_AIRPORT"
```

```r
colnames(df)[colnames(df) == "DESTINATION_AIRPORT.y"] = "DESTINATION_AIRPORT"
colnames(df)[colnames(df) == "AIRLINE.y"] = "AIRLINE"

#Drop Flights w/ No Departure Time
df <- df[!(is.na(df$DEPARTURE_TIME) | df$DEPARTURE_TIME == ""), ]

#Convert Month and Day Of Week
df$MONTH[df$MONTH == 1] <- "January"
df$DAY_OF_WEEK[df$DAY_OF_WEEK == 1] <- "Monday"
df$DAY_OF_WEEK[df$DAY_OF_WEEK == 2] <- "Tuesday"
df$DAY_OF_WEEK[df$DAY_OF_WEEK == 3] <- "Wednesday"
df$DAY_OF_WEEK[df$DAY_OF_WEEK == 4] <- "Thursday"
df$DAY_OF_WEEK[df$DAY_OF_WEEK == 5] <- "Friday"
df$DAY_OF_WEEK[df$DAY_OF_WEEK == 6] <- "Saturday"
df$DAY_OF_WEEK[df$DAY_OF_WEEK == 7] <- "Sunday"

#Add Columns For Additional Context
df <- df %>%
  mutate(DELAYED_1 = ifelse(df$DEPARTURE_DELAY < 60 & df$DEPARTURE_DELAY > 0,
1, 0)) %>%
  mutate(DELAYED_2 = ifelse(df$DEPARTURE_DELAY <= 120 & df$DEPARTURE_DELAY >=
60, 1, 0)) %>%
  mutate(DELAYED_3 = ifelse(df$DEPARTURE_DELAY <= 180 & df$DEPARTURE_DELAY >
120, 1, 0)) %>%
  mutate(DELAYED_4 = ifelse(df$DEPARTURE_DELAY > 180, 1, 0)) %>%
  mutate(is_DELAYED = ifelse(df$DEPARTURE_DELAY > 0, 1, 0)) %>%
  mutate(ON_TIME = ifelse(df$DEPARTURE_DELAY > 0, 0, 1)) %>%
  mutate(region = state.name[match(ORIGIN_STATE, state.abb)])

###GRAPHS###

#On-time vs Delayed
paste0("Flights in January 2015: ", format(nrow(df), big.mark=",", scientific
=FALSE))

## [1] "Flights in January 2015: 458,311"

paste0("# of Those Delayed: ", format(sum(df$is_DELAYED), big.mark=",", scien
tific=FALSE), " (",round((sum(df$is_DELAYED)/nrow(df))*100, 2),"%)")

## [1] "# of Those Delayed: 176,627 (38.54%)"

#Departure/Arrival Correlation
#FIGURE 1
pairs(df[c("DEPARTURE_DELAY", "ARRIVAL_DELAY")])

cor(df[c("DEPARTURE_DELAY", "ARRIVAL_DELAY")], use = "complete.obs")

##                 DEPARTURE_DELAY ARRIVAL_DELAY
## DEPARTURE_DELAY       1.000000      0.938116
## ARRIVAL_DELAY         0.938116      1.000000
```

```r
#Delays by Day
month_delayed_1 <- df %>%
  group_by(DAY) %>%
  summarise(DELAYED_1 = sum(DELAYED_1))
month_delayed_2 <- df %>%
  group_by(DAY) %>%
  summarise(DELAYED_2 = sum(DELAYED_2))
month_delayed_3 <- df %>%
  group_by(DAY) %>%
  summarise(DELAYED_3 = sum(DELAYED_3))
month_delayed_4 <- df %>%
  group_by(DAY) %>%
  summarise(DELAYED_4 = sum(DELAYED_4))

month_delays <- month_delayed_1 %>%
  left_join(month_delayed_2, by = "DAY")
month_delays <- month_delays %>%
  left_join(month_delayed_3, by = "DAY")
month_delays <- month_delays %>%
  left_join(month_delayed_4, by = "DAY")

#FIGURE 2
ggplot(month_delays, aes(x = DAY)) +
  geom_line(aes(y = DELAYED_1, color = "< 1 Hour") ) +
  geom_line(aes(y = DELAYED_2, color = "1-2 Hours") ) +
  geom_line(aes(y = DELAYED_3, color = "2-3 Hours") ) +
  geom_line(aes(y = DELAYED_4, color = "> 3 Hours") ) +
  labs(x = "Day of Month",
       y = "# of Delays",
       title = "# of Delays by Day in January") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(labels = comma) +
  guides(color = guide_legend(title = "Length"))

#Delays by Day Of Week
day_delays <- df %>%
  select(DAY_OF_WEEK, DELAYED_1, DELAYED_2, DELAYED_3, DELAYED_4) %>%
  pivot_longer(-c(DAY_OF_WEEK), names_to = "Length", values_to = "Value")

day_delays <- day_delays %>%
  group_by(Length, DAY_OF_WEEK) %>%
  summarise(total = sum(Value))

## `summarise()` has grouped output by 'Length'. You can override using the
## `.groups` argument.

day_delays$Length <- factor(day_delays$Length, levels = c("DELAYED_1", "DELAY
ED_2", "DELAYED_3", "DELAYED_4"))

#FIGURE 3
```

```r
ggplot(day_delays, aes(fill = Length, y = total, x = reorder(DAY_OF_WEEK, -to
tal))) +
  geom_bar(position = "stack", stat = "identity") +
  labs(x = "Day of Week",
       y = "# of Delays",
       title = "# of Delays by Day of Week") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_discrete(labels = c("< 1 Hour", "1-2 Hours", "2-3 Hours", "> 3 H
ours")) +
  scale_y_continuous(labels = comma)

#Delays by AIRLINE
airline_delay <- df %>%
  select(AIRLINE, DELAYED_1, DELAYED_2, DELAYED_3, DELAYED_4) %>%
  pivot_longer(-c(AIRLINE), names_to = "Length", values_to = "Value")

airline_delay <- airline_delay %>%
  group_by(Length, AIRLINE) %>%
  summarise(total = sum(Value))

## `summarise()` has grouped output by 'Length'. You can override using the
## `.groups` argument.

airline_totals <- airline_delay %>%
  group_by(AIRLINE) %>%
  summarise(whole = sum(total))

airline_delay <- airline_delay %>%
  left_join(airline_totals,by="AIRLINE") %>%
  mutate(percent = total/whole)

airline_delay$Length <- factor(airline_delay$Length, levels = c("DELAYED_1",
"DELAYED_2", "DELAYED_3", "DELAYED_4"))

#FIGURE 4
ggplot(airline_delay, aes(fill = Length, y = percent, x = AIRLINE)) +
  geom_bar(position = "stack", stat = "identity") +
  labs(x = "Airline",
       y = "Percent",
       title = "Distribution of Delay Length by Airline") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_discrete(labels = c("< 1 Hour", "1-2 Hours", "2-3 Hours", "> 3 H
ours")) +
  coord_flip()

#Linked Micro Map
statePolys <- create_map_table(USstates, IDcolumn = "ST")

mapDF <- df %>%
  select(region, ORIGIN_STATE, is_DELAYED, ON_TIME) %>%
```

```
  group_by(ORIGIN_STATE, region) %>%
  summarise(is_DELAYED = sum(is_DELAYED), ON_TIME = sum(ON_TIME))

## `summarise()` has grouped output by 'ORIGIN_STATE'. You can override using
the
## `.groups` argument.

mapDF <- mapDF[!(is.na(mapDF$region) | mapDF$region == ""), ]

#FIGURE 5
mmplot(stat.data = mapDF, map.data = statePolys,
       panel.types = c("map", "dot_legend", "labels", "dot", "dot"),
       panel.data = list(NA, NA, "region", "is_DELAYED", "ON_TIME"),
       map.link = c("ORIGIN_STATE", "ID"),
       ord.by = "is_DELAYED",
       grouping = 5,
       median.row = F,
       plot.height = 4,
       plot.width = 10,
       colors = c("red", "green", "blue", "orange", "yellow"),
       panel.att = list(list(1, header = "Selected (Highlighted)\nPrevious (G
ray)", panel.width = .3),
                        list(2, panel.width = .4),
                        list(3, header = "State", align = "left", panel.width
= .3, text.size = .9),
                        list(4, header = "# of Flights\n Delayed",
                             graph.bgcolor = "lightgray", point.size = 1,
                             xaxis.title = "# of Flights\n(Thousands)", xaxis
.ticks = list(0, 5000, 10000, 15000, 20000, 25000), xaxis.labels = list("0",
"5", "10", "15", "20", "25")),
                        list(5, header = "# of Flights\n On-Time",
                             graph.bgcolor = "lightgray", point.size = 1,
                             xaxis.title = "# of Flights\n(Thousands)", xaxis
.ticks = list(0, 5000, 10000, 15000, 20000, 25000, 30000, 35000), xaxis.label
s = list("0", "5", "10", "15", "20", "25", "30", "35"))))

#Delays by Origin Airport
top_10 <- df %>%
  group_by(ORIGIN_AIRPORT) %>%
  summarise(total = sum(is_DELAYED)) %>%
  top_n(10)

## Selecting by total

top_10 <- as.list(top_10$ORIGIN_AIRPORT)

airport_delays <- df %>%
  select(ORIGIN_AIRPORT, DELAYED_1, DELAYED_2, DELAYED_3, DELAYED_4) %>%
  pivot_longer(-c(ORIGIN_AIRPORT), names_to = "Length", values_to = "Value")

airport_delays <- airport_delays %>%
```

```
  group_by(Length, ORIGIN_AIRPORT) %>%
  summarise(total = sum(Value)) %>%
  filter(ORIGIN_AIRPORT %in% top_10)

## `summarise()` has grouped output by 'Length'. You can override using the
## `.groups` argument.

airport_delays$Length <- factor(airport_delays$Length, levels=c("DELAYED_1",
"DELAYED_2", "DELAYED_3", "DELAYED_4"))

#FIGURE 6
ggplot(airport_delays, aes(fill = Length, y = total, x = reorder(ORIGIN_AIRPO
RT, total))) +
  geom_bar(position = "stack", stat = "identity") +
  labs(x = "Airport",
       y = "# of Delays",
       title = "Top 10 Delayed Airports") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_discrete(labels = c("< 1 Hour", "1-2 Hours", "2-3 Hours", "> 3 H
ours")) +
  scale_y_continuous(labels = comma) +
  coord_flip()

###MODEL###

#Random Forest
model <- df %>%
  select(is_DELAYED, ORIGIN_STATE, ORIGIN_AIRPORT, AIRLINE, DAY_OF_WEEK, DAY)

model <- model[!(is.na(model$ORIGIN_STATE) | model$ORIGIN_STATE == ""), ]

sample <- sample(c(TRUE, FALSE), nrow(model), replace = TRUE, prob = c(0.7, 0
.3))
train  <- model[sample, ]
test   <- model[!sample, ]

train <- train[sample(nrow(train), 5000), ]
test <- test[sample(nrow(test), 5000), ]

train$is_DELAYED <- as.factor(train$is_DELAYED)

flights.rf <- randomForest(is_DELAYED ~ .,
                           data = train,
                           importance = TRUE)

#FIGURE 7
plot(flights.rf)

#FIGURE 8
varImpPlot(flights.rf)
```

```r
print(flights.rf)
```

```
##
## Call:
##  randomForest(formula = is_DELAYED ~ ., data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 35.34%
## Confusion matrix:
##      0   1 class.error
## 0 2415 669   0.2169261
## 1 1098 818   0.5730689
```

```r
pred_DELAY <- predict(flights.rf, newdata = test)

confusionMatrix(table(pred_DELAY,test$is_DELAYED))
```

```
## Confusion Matrix and Statistics
##
##
## pred_DELAY    0    1
##          0 2471 1149
##          1  629  751
##
##                Accuracy : 0.6444
##                  95% CI : (0.631, 0.6577)
##     No Information Rate : 0.62
##     P-Value [Acc > NIR] : 0.0001895
##
##                   Kappa : 0.2031
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7971
##             Specificity : 0.3953
##          Pos Pred Value : 0.6826
##          Neg Pred Value : 0.5442
##              Prevalence : 0.6200
##          Detection Rate : 0.4942
##    Detection Prevalence : 0.7240
##       Balanced Accuracy : 0.5962
##
##        'Positive' Class : 0
##
```