1 Table of Contents

2 Data Exploration:

Understanding the data set was the initial challenge that I faced. Several questions popped up with respect to the new terms and it took time for me to search the internet to get the exact meanings of the terms used.

We have details of airline carriers and the flights it operated since January across various airports in the United States. In addition to this the data had details about the flight delays, cancellations and several features describing the travel time.

I used Tableau to do some initial explorations of data and the below links have few dashboards that I have created on this data.

2.1 Dashboard - Flight Trends:

This dashboard has two sheets that show the total trips or the total counts of flights that happened in the month of January.

https://public.tableau.com/views/FlightTrends/Dashboard-FlightTrends?:embed=y&:display_count=yes&:showTabs=y

Trend-by day:

This sheet shows us how the total count of flights changed over the month. They clearly signify a trend on a weekly basis.

Trend-by day-by airline/carrier:

This sheet is similar to the previous one but shows us the trend in number of flights that are operated by each of the unique carriers over the month of January. Clearly each of the major carriers show a trend and looks like it captures the market behavior.

Summary:

The two sheets in this dashboard gives us an idea as to how the operations of flights across the United States happened in the month of January. Clearly the number of flights decreases or goes down as we approach the weekends and is the lowest on Saturdays.

The second sheet gives us a clear picture of major carriers and “WN” (Southwest) dominates the airline industry in the month January as far the number of flights operated is concerned.

2.2 Dashboard – Carrier Stats

This dashboard gives us insights on each of the carriers, it has three sheets that gives us an overview of how each of them performed in January.

https://public.tableau.com/views/CarrierStats/Dashboard-CarrierStats?:embed=y&:display_count=yes&:showTabs=y

Taxis-by carrier:

This is a simple sheet that gives the count of the flights operated by each of the carriers. It can help us figure out the market leaders in the airline industry. Clearly WN or the southwest airlines outplayed its competitors by operating thrice the number of flights operated by its competitors.

Cancellations-By Carrier:

This sheet shows the number of cancelled flights grouped by each carrier. Surprisingly WN holds the second spot below AA which shows its better in operating performance. The number of flights operated by WN is clearly more than others but its cancellations fall below AA which is an identifier indicating its service quality. We will have to dig deep and analyze the other reasons for cancellation too before conclusion.

Delays-by Carrier:

This sheet gives us better understanding of the various parameters that cause delays which are in turn grouped by carriers. We get better insights from the graphs and it helps us compare the carriers by their average delays. Assessing performance by delays is one valuable take-away from this sheet.

Summary:

This dashboard gives us better understanding of how each of the unique carriers operate and help us measure performance over the month of January. Several comparative studies could be done and the results are quite obvious with this visualization.

2.3 Dashboard-Busiest Airport

A simple dashboard that gives us information on airports visually. It has three sheets giving us insights on different airports across Unites States

https://public.tableau.com/views/BusiestAirport/Dashboard-BusiestAirport?:embed=y&:display_count=yes&:showTabs=y

Busiest Airport – By day

An intuitive display of which airport was busy in January, we can scroll through the right to view the trend throughout the month. Looks like ATL (Atlanta) was busy almost all days.

Origin-Density.

This is a different plot which could be used to visualize the flight density across the airports. Clearly ATL has the biggest bubble indicating that it is the busiest having the maximum count of flights operated in January followed by ORD, LAX, DFW.

Busiest Airport-By month

This sheet again gives us information the airport and the count of flights operated. Kind of a skewed plot with lesser visibility but we find some details by hovering over the bars.

Summary:

This dashboard helps us visualize the operations of different airports across United States in the month of January and helps asses their busyness

3 Data Cleansing/Manipulation & Summary Statistics:

Used DPLYR to clean data and get some insights on the distribution.

Problem Statement:

The idea is to run a logistic regression model to predict the chances that a flight may get delayed. The use is that, a customer making a reservation could get some information of the probability that the flight may get delayed based on some features.

Derived Variables:

'total_delay' & 'delayed'.

The dataset shows several fields indicating the delays that could happen. The idea is to study about the delays. Created a secondary variable which is the sum of all the delays and named it 'total_delay'. I then created a label named 'delayed' if the 'total_delay' was greater than zero.

```
#create a secondary variable by using mutate function of dplyr to get the total delay

data <- mutate(data, total_delay = (CARRIER_DELAY + WEATHER_DELAY + NAS_DELAY + SECURITY_DELAY +LATE_AIRCRAFT_DELAY))

#function to label whether the flight is delayed or not

cal_delay = function(x) {
  if(is.na(x)){
               y = 0
            }
  else    {
              if(x > 0) {
                y = 1
```

```
                }
            }
    return(y)
}

#mutate to create a new variable DELAYED, the label that is to be used as the
dependant variable

data <- mutate(data, delayed = unlist(lapply(data$total_delay, cal_delay)))
```

Obtaining the day of week:

Used the **lubridate** package to obtain the day of week which could be a significant contributor to the model.

```
#Converting the FL_DATE to date type using lubridate library

data$FL_DATE <- mdy(data$FL_DATE)

#getting the day of week using lubridate

data <- mutate(data, day_of_week = wday(data$FL_DATE, label=TRUE))
```

Summary Statistics:

Grouped data by day of week to see how the delays are. Got the total and mean delay.

```
attach(data)
group <- group_by(data, day_of_week)

#gives the total delays grouped by week day, this gives us a picture of
delays on each day for January

sum <- summarise(group, sum=sum(delayed))

#arrange the result in descending order
arrange <- arrange(sum, desc(sum))

#to get the mean of the delays by using summarise of dplyr

mean <- summarise(group, mean=mean(delayed)) %>% arrange(desc(mean))
```

```
> group <- group_by(data, day_of_week)
> sum <- summarise(group, sum=sum(delayed))
> arrange <- arrange(sum, desc(sum))
> arrange
Source: local data frame [7 x 2]

  day_of_week   sum
       (fctr) (int)
1         Fri 13003
2         Sun 12739
3         Mon 10912
4       Thurs  8926
5        Tues  8570
6         Sat  8409
7         Wed  8323
>
```

```
> mean <- summarise(group, mean=mean(delayed)) %>% arrange(desc(mean))
> mean
Source: local data frame [7 x 2]

  day_of_week      mean
       (fctr)     (dbl)
1         Sun 0.1803012
2         Mon 0.1788032
3         Fri 0.1753891
4        Tues 0.1470664
5       Thurs 0.1462224
6         Wed 0.1409818
7         Sat 0.1363902
```

The above two measure show that it is best to travel on Saturdays followed and then in mid weeks.

Similarly use DPLYR to group by carriers and get the total and mean delay across each carrier:

```
#lets try grouping by airlines to get some summary statistics

group_car <- group_by(data, CARRIER)

#summarise and arrange to get some insights

summary <- summarise(group_car, total_delay_carrier=sum(delayed)) %>%
arrange(desc(total_delay_carrier))

#now for the mean delays which gives insights as to which airlines is good
wrt delays
summary_mean <- summarise(group_car, avg_delay_carrier=mean(delayed)) %>%
arrange(desc(avg_delay_carrier))
```

```
> group_car <- group_by(data, CARRIER)
> summary <- summarise(group_car, total_delay_carrier=sum(delayed)) %>% a
)
> summary
Source: local data frame [12 x 2]

   CARRIER total_delay_carrier
    (fctr)               (int)
1       WN               14103
2       AA               12444
3       OO                9499
4       DL                9234
5       EV                6375
6       B6                6098
7       UA                5571
8       NK                3210
9       AS                1655
10      VX                1205
11      F9                 978
12      HA                 510
```

```
> summary_mean <- summarise(group_car, avg_delay_carrier=mean(delayed)) %>% arr
r))
> summary_mean
Source: local data frame [12 x 2]

   CARRIER avg_delay_carrier
    (fctr)             (dbl)
1       NK        0.29057663
2       B6        0.26492310
3       VX        0.22381129
4       OO        0.19947920
5       AA        0.16464673
6       EV        0.15189421
7       UA        0.14011217
8       F9        0.13776588
9       WN        0.13540527
10      DL        0.13246116
11      AS        0.11650827
12      HA        0.08122312
> |
```

The above measure show that the carrier NK has the highest mean delay, thus choosing this airlines may result in high chances of delays.

Created another secondary variable to bucket the departure times and classified them into 8 buckets.

I chose to use **Departure Time**, and there existed a confusion as to which time to use, CRS or Local Time, I then decided to go with Local time, as different reservation systems might have their servers hosted on different time zones.

Local departure times logically seems to be a better choice and could better generalize.

```
#function to bucket the departure time to Rush hour, Business Hours, Mid-
day, lazy-hour, Evening-Rush, night, mid-night, morning

split_hour <- function(x) {
                if(x > 0000 & x <= 0300) {
                    y = "Mid-night"
                }
```

```
              else if(x > 0300 & x <= 0600) {
                y = "Morning"
              }
              else if(x > 0600 & x <= 0900) {
                y = "Rush-Hour"
              }
              else if(x > 0900 & x <= 1200){
                y = "Business-hour"
              }
              else if(x > 1200 & x <= 1500){
                y = "Mid-day"
              }
              else if(x > 1500 & x <= 1800){
                y = "lazy-hour"
              }
              else if(x > 1800 & x <= 2100){
                y = "Evening-Rush"
              }
              else {
                y = "Night"
              }
      return(y)

}

#removing the cancelled flights as we assume that delay cannot exist when a flight gets cancelled
data <- filter(data, !is.na(data$DEP_TIME))

#removed 11473 rows which had null in departure time as the flights were cancelled
str(data)

#creating the new secondary derived variable to bucket the departure time as described above
data <- mutate(data, hour = sapply(data$DEP_TIME, split_hour, USE.NAMES=FALSE))
```

4 Model Building

I am planning to build a logistic regression model to predict whether a flight would be delayed based on few features in the data set: here "delayed" is the dependent variable and day_of_week, hour, distance, origin, destination, carrier as the independent predictors.

As mentioned above, I selected the features from the cleaned dataset.

```
#selecting the required variables from the cleaned dataset
```

```
final <- select(data, CARRIER, ORIGIN, DEST, DISTANCE, delayed,
day_of_week, hour)

#chcking for nulls in the "final" dataset - result is false which confirms
no null
any(is.na(final))

#converting "hour" and “delayed” to factor

final$hour <- as.factor(final$hour)
final$delayed <- as.factor(final$delayed)
```

4.1 Logistic Regression Model:

Without any test/train split, just tried to check if my computer has the capacity to work on such a big dataset:

```
#given the timeframe of 2 to 3 hours, I am unable to do further checks to
the dateset
attach(final)

logistic_delay <- glm(delayed ~ CARRIER + ORIGIN + DEST + DISTANCE +
day_of_week + hour, data = final, family = "binomial")

summary(logistic_delay)
```

The above model failed to build on my system due to memory issues:

```
Console C:/Sriram/Resumes/Full time/The advisory Board/data/
> str(data)
'data.frame': 434354 obs. of  8 variables:
 $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ CARRIER    : Factor w/ 12 levels "AA","AS","B6",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ ORIGIN     : Factor w/ 294 levels "ABE","ABQ","ABR",..: 82 257 149 263 209 220 159 209 176 76 ...
 $ DEST       : Factor w/ 294 levels "ABE","ABQ","ABR",..: 77 149 257 209 263 77 75 278 209 185 ...
 $ DISTANCE   : int  986 2422 2422 1829 1829 868 2311 1012 1005 1709 ...
 $ delayed    : int  1 0 0 0 0 0 1 0 0 0 ...
 $ day_of_week: Factor w/ 7 levels "Fri","Mon","Sat",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ hour       : Factor w/ 8 levels "Business-hour",..: 3 8 2 4 3 1 4 3 4 1 ...
> attach(data)
> logistic_delay <- glm(delayed ~ CARRIER + ORIGIN + DEST + DISTANCE + day_of_week + hour, data = data, family = "bin
omial")
Error: cannot allocate vector of size 2.0 Gb
In addition: Warning messages:
1: In glm.fit(x = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  :
  Reached total allocation of 8100Mb: see help(memory.size)
2: In glm.fit(x = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  :
  Reached total allocation of 8100Mb: see help(memory.size)
3: In glm.fit(x = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  :
  Reached total allocation of 8100Mb: see help(memory.size)
4: In glm.fit(x = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  :
  Reached total allocation of 8100Mb: see help(memory.size)
```

I then decided to include the major ten airport in the United states and work on them by selecting a sample out of them and building a model.

```
#due to memory error, grouping by origin aiport and checking for
distribution

group <- group_by(data, ORIGIN)
summary1 <- summarise(group, count = n()) %>% arrange(desc(count))

#for our analysis, we will consider the 10 major airports and pick 1000
records randomly from them

final_filter <- filter(final, ORIGIN %in%
c("ATL","ORD","DEN","DFW","LAX","SFO","PHX","LAS","IAH","MCO"))

#now picking random samples from this data
set.seed(100)
final_sample <- sample_n(final_filter, 15000)

#checking for delaye flight counts from individual origin airports

group_by(final_sample, ORIGIN) %>% summarise(count = sum(delayed)) %>%
arrange(desc(count))
```

```
> summary1 <- summarise(group, count = n()) %>% arrange(desc(count))
> summary1
Source: local data frame [294 x 2]

   ORIGIN count
   (fctr) (int)
1     ATL 29870
2     ORD 18610
3     DEN 17519
4     DFW 16565
5     LAX 16427
6     SFO 13207
7     PHX 13024
8     LAS 12246
9     IAH 11660
10    MCO 10739
..    ...   ...
```

```
> group_by(final_sample, ORIGIN) %>% summarise(count = sum(delayed)) %>% arrange(desc(count))
Source: local data frame [10 x 2]

   ORIGIN count
   (fctr) (int)
1     ORD   392
2     ATL   366
3     LAX   323
4     SFO   291
5     DEN   270
6     DFW   216
7     LAS   215
8     MCO   208
9     PHX   198
10    IAH   147
>
```

Split – Test/Train

```
library("caTools")
sample = sample.split(final_sample$delayed, SplitRatio = .70)
train = subset(final_sample, sample == TRUE)
test = subset(final_sample, sample == FALSE)
```

Final Model

```
#logistic with this refined data

attach(train)

logistic_delay <- glm(delayed ~ CARRIER + ORIGIN + DISTANCE + day_of_week +
hour, data = train, family = "binomial")

summary(logistic_delay)
```

5 Evaluation:

Prediction on the test data:

```
k <- predict.glm(logistic_delay,test,type="response")
```

Confusion Matrix based on various cut-off's

```
Library(caret)
cut.off <- 0.2;
pred.chargeoff <- (k > cut.off);
confusionMatrix(test$delayed,as.numeric(pred.chargeoff))

cut.off <- 0.4;
```

```
pred.chargeoff <- (k > cut.off);
confusionMatrix(test$delayed,as.numeric(pred.chargeoff))

cut.off <- 0.35;
pred.chargeoff <- (k > cut.off);
confusionMatrix(test$delayed,as.numeric(pred.chargeoff))
```

```
> k <- predict.glm(logistic_delay,test,type="response")
> cut.off <- 0.2;
> pred.chargeoff <- (k > cut.off);
> confusionMatrix(test$delayed,as.numeric(pred.chargeoff))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2691 1021
         1  409  379

               Accuracy : 0.6822
                 95% CI : (0.6684, 0.6958)
    No Information Rate : 0.6889
    P-Value [Acc > NIR] : 0.837
```

```
> cut.off <- 0.3;
> pred.chargeoff <- (k > cut.off);
> confusionMatrix(test$delayed,as.numeric(pred.chargeoff))
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3546  166
         1  695   93

               Accuracy : 0.8087
                 95% CI : (0.7969, 0.8201)
    No Information Rate : 0.9424
    P-Value [Acc > NIR] : 1
```