Consulting Project for Aviation Route Management

Using free data from the Department of Transportation to find most delayed routes and recommend routes to add flight volume using Databricks and Tableau

Source: https://pixabay.com/photos/airport-transport-woman-girl-2373727/

Technical Report by: Thirumurugan Vinayagam
Project by: Namita Ramesh, Arjun Rao, Prathik Ullur, Thirumurugan Vinayagam, Matt Zlotnik

Abstract

The airline visualization project uses free data from Transtats, a subsidiary of the US Department of Transportation, and OpenFlights data to find the most delayed routes and routes to add flight volume for the four major airlines in the United States. The project uses only domestic data with a calculated weighted average of total delay as a performance metric for each flight. We use Spark for handling our large dataset and Tableau for visualization. Our objective for the project was to create a value-add dashboard, using free public data, for pricing and revenue management managers, to help focus resources on low-performing routes, and reward high performing routes with potentially more traffic.

Business Problem

With our interest in aviation, and lacking actual data from specific airlines, for legal reasons, the team decided to find online data on airline performance in the United States. We found and worked on the airline flight carrier on-time performance data set found on the Transtats website, which is sponsored by the US Department of Transportation. Using raw data, we were tasked on finding business insights in the data.

Our initial inclination was to find the most delayed routes based on a given airline to give a focus area for the operations and route management teams at an airline. Based on the most delayed routes overall these departments could spend their limited resources on improving the total delay for a given route.

The other objective for the team was to create a recommendation map for airlines to add flight volume. Our model for the routes to add flight volume will be discussed later in the paper. Given this dashboard the pricing and route management department would be able to create demand forecasts and reallocate aircraft to increase their brands presence on a route.

Although the team researched other clients, such as Kayak and flight pricing, the technical paper will only address my specific contribution to the project, namely the Spark SQL code to find the most delayed routes and routes to increase flight volume, as well as the Tableau dashboard for visualizing these findings.

The Dataset

The data from Transtats has thirty-five attributes, however of those we chose only six attributes to focus on for our project: the reporting airline, the airline IATA code, the origin and destination airports, and the delay of both take-off and landing. The team chose to concentrate on recent data from August 2016 to August 2019. Each month of data had roughly three hundred thousand rows, with the total dataset having around twenty million rows. Our subset of the overall dataset, which is thirty-five gigabytes (GBs) of data, was around five GBs.

The OpenFlights data set has fourteen attributes of which we only used three: the IATA code, latitude, and longitude. The only use for this airline sheet was to add location data for each airport which will be used for mapping data onto the Tableau dashboard. Links for both the Transtats and the OpenFlights data respectively has been included in the references section.

Data Cleanup

The Transtats website forces individuals to download data for one month at a time. Since we were looking at data from August 2016 to August 2019, we downloaded thirty-seven individual csv files. I then wrote a Python script that combined each of the csv files into one file for each year.

```
import pandas as pd          #Uses Pandas for dataframe import and concat
numbers= list(range(30,38)) #picks index for dates in a certain year, example is for 2019
dates=[]                     #Empty list to append dates
Full_data=[]                 #Empty list to append each csv file
count=0                      #Starting the counter for a progress bar
for i in numbers:
    data = pd.read_csv("~/Desktop/DBM_Project/Data/90229638_T_ONTIME_REPORTING-{0}.csv".format(i))
                                 #Reads each csv in the given range of numbers
    Full_data.append(data)       #Appends the current csv to the list of previous csvs
    dates.append(data['FL_DATE'][0])  #Appends the date of the first flight of the csv
    count+=1                     #Made a counter for a progress bar
    print(count)                 #Printing count so we can see the progress of the loop
#For loop will take all csvs from the local computer and
Full_data2019=pd.concat(Full_data)    #Combines all the pandas datasets from the Full Data list into one csv
Full_data2019.to_csv("2019data.csv",index=False)   #Downloads the Full Data for a given year to your local computer
print(dates)                  #Prints the dates of the data just to make sure we have all the data for a given year
```

My Python Script that combines the thirty-seven csv files into four files.

After the data was combined into yearly data, I imported the four csv files into Databricks using Spark SQL. I used the union function to combine the four csv files into one file for all four years.

For the OpenFlights data, I immediately imported the csv into Databricks again using Spark SQL. I converted the file into a Pandas data frame to make merging easier.

Aviation Metrics

Most Delayed Routes

For the most delayed route the key performance metric we used was the Weighted Average Total Delay for a given route. There was a sixty percent weight for Arrival Delay and a forty percent weight for departure delay. The reasoning behind these weights is that arrival delay for a flight means the amount of time that a passenger is late in total, since the arrival delay is calculated as the actual arrival time minus the scheduled arrival delay. Meanwhile, the departure delay is able to be reduced by either flying faster than normal, flying a different route with alternate waypoints, or other factors. Also, since airlines today generally overestimate the total flight time as a buffer for operations and passengers alike, the buffer will reduce the overall impact of the departure delay for a flight.

Flight Volume Recommendation

The two key performance metrics for the route recommendations was that a given airline was already flying the route and the total delay metric, from the most delayed routes, divided by the flight volume. The route recommendations were made only if the given airline already flew the route, since adding a brand-new route requires intensive demand forecasting and network compatibility, including aircraft availability and connection to the existing network. Also, we chose to add flight volume to a route where an airline had the lowest total delay for the route divided by the number of flights they currently fly to that destination.

Data Manipulation

Most Delayed Routes

The code to find the most delayed routes was extremely simple. By using Spark SQL, I selected the airline, the origin, destination, and created the total delay metric through this statement, ROUND(AVG(.6*ARR_DELAY+.4*DEP_DELAY),2). I ordered by the airline, origin, and destination and sorted by the total delay metric with the highest total delay first. After converting this Spark data frame to a Pandas data frame, I filled all of the NA values with zero. This gave a Pandas data frame with the most delayed routes for a given airline. The next step was to merge this data frame with the previous OpenFlights data frame by using the IATA column with the origin and destination. This gave a data frame with the airline, an origin, and destination, the total delay on the route, and the latitude and longitude for both the origin and destination airports.

Flight Volume Recommendation

As the more complicated project the flight volume recommendation took many steps. First, I made the delay per flight metric by dividing the total delay by the flight volume for a given route. Then by using a pivot table, which indexed by a route and had each column as an airline, I filled the delay per flight metric as the values in the pivot table. After making a column for a

row-wise addition, I filtered for all routes that had more than one airline flying that route. I then created a filter for finding the best airline for a given route, then merged that data with the location data just like the most delayed routes merge. The final csv gave the airline, the origin, the destination, and the latitude and longitude of those airports.

Results and Visualizations

From the eight csv files for both parts of this project, I created two csv files, one for most delayed routes and one for the flight volume dashboard. Since I learned Tableau especially for this project, I had to manually change the csv's to make the visualization easier.

Most Delayed Routes

For the most delayed routes, I took the top ten most delayed routes from each different csv. I made one csv with five columns: the airline code, the airport, the latitude, the longitude, and the path – one if the airport is an origin and two if it is a destination. I created the following dashboard in Tableau using the path as the network connection and created a filter for each airline, which is not shown in the image. The thickness of a line for a given route is sized by the total delay of the route, which is not easily seen in the image.

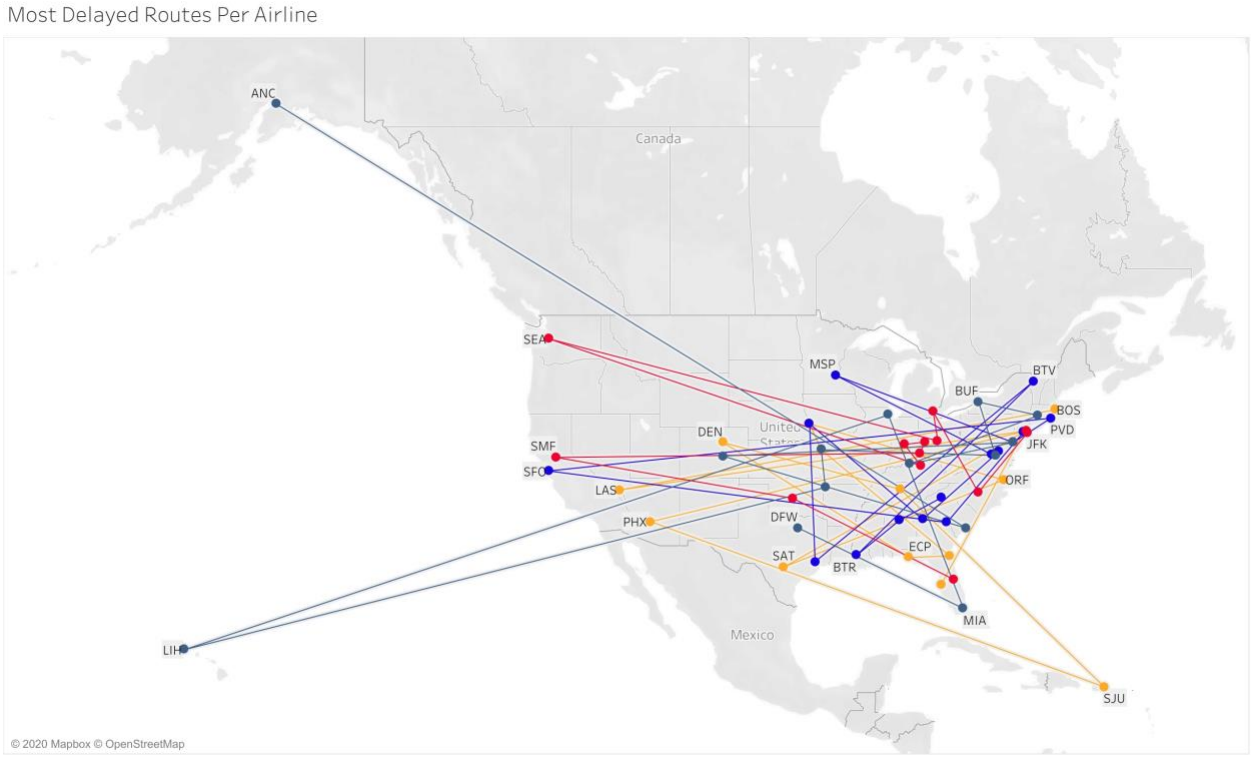

Tableau Dashboard using Tableau Public for the Most Delayed Routes.

Flight Volume Recommendation

I made the same changes for the flight volume csvs. I made one csv with five columns: the airline code, the airport, the latitude, the longitude, and the path – one if the airport is an origin

and two if it is a destination. I created the following dashboard in Tableau using the path as the network connection and created a filter for each airline.

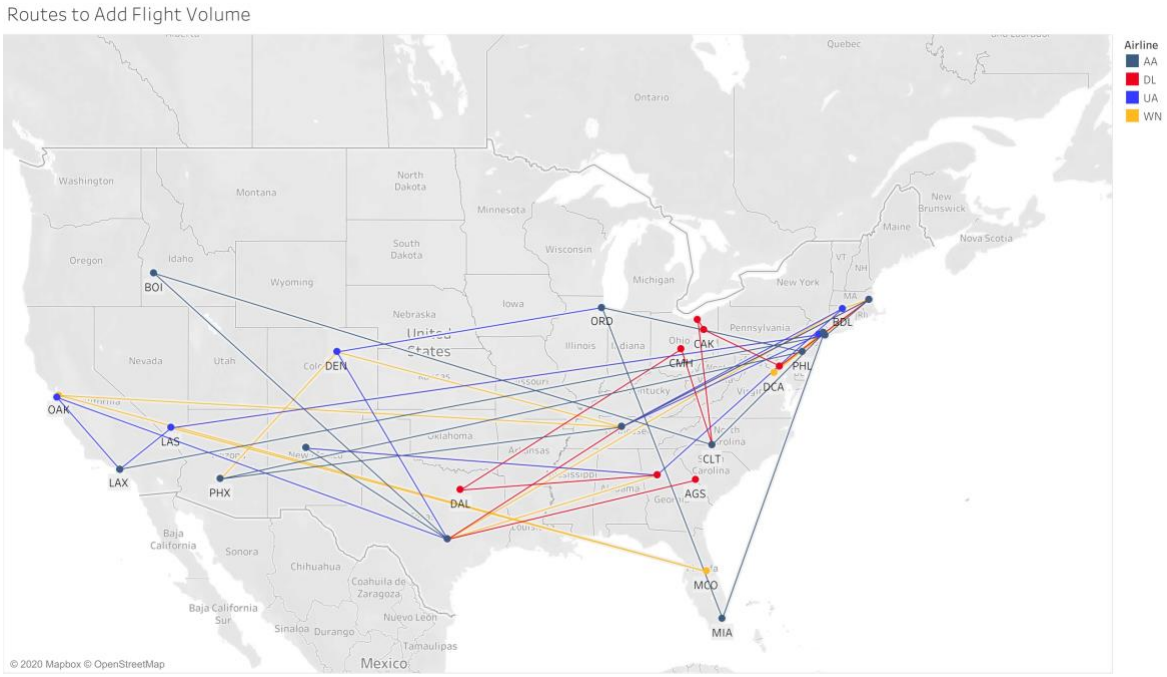

Tableau Dashboard using Tableau Public for the Flight Volume Recommendation.

References

https://www.transtats.bts.gov/Tables.asp?DB_ID=120&DB_Name=Airline%20On-Time%20Performance%20Data&DB_Short_Name=On-Time

https://raw.githubusercontent.com/jpatokal/openflights/master/data/airports.dat