Flight Cost Analysis and Application

Anderson Matt, John Cai, Johnson Rojae, Karnauch Andrey, and Kramer Matthew

Abstract— **Proposal for an application that helps users determine which airport, within a certain driving distance, is cheapest to fly out of to reach their destination.**

I. Objective (Research Question)

To create a web-based tool that advises users on lowest price point for airline travel from a user-defined region to the city of destination. The user-defined region is an area centered on the users home city that may include multiple airports. The goal is to determine which is the most cost effective airport to depart from while accounting for travel time/costs to the airports that are not the most proximal. Thus allowing users to effectively plan budget travel based around their location, willingness to drive, travel dates and destination.

II. Motivation

This project is motivated by a gap in existing algorithmic tools for travelers. Currently, multiple resources exist for travelers to determine which airline is most cost-effective for travel between a set departure and destination airport, but none can be found by the research team that generalize further to allow dynamic departure airports. This is desirable as there can be significant price differences between neighboring airports. For example, searching for flights from Knoxville to Boston, we see a variation of over $120 depending on if you depart from Knoxville, Chattanooga, or Nashville. Clearly, if based in Knoxville it is most convenient to depart from this airport, but if driving to Nashville costs $30 the resulting savings is $90, a non-trivial amount to students and budget travelers. Thus this tool is motivated by and directed at facilitating reductions in travel cost for budget travelers, who may be unable to travel without these reductions.

III. Data to be Used

A. Historical Data

We will be using historical flight data provided by the Bureau of Transportation Statistics (BTS)[1] as our core data for this application. The two tables that provide the relevant information for our application are **DB1BTicket** and **DB1BMarket** (see link [1] in the Reference section below). Both of the tables have "no missing values or some missing values," where <u>some</u> is less than .005% according to the BTS table description of each table. This means that our data sets will likely not have any columns with NULL values. The data can be downloaded from the BTS website in increments of **annual quarters**. So if we want to obtain all historical flight data from 2015-2018 using these two tables, we need to download **40** different files (5 years * 4 quarters each year * 2 tables). Each of these files is a CSV file, and the *DB1BMarket* quarterly tables are an average of 1.5GB each while the *DB1BMarket* quarterly tables are an average of 500MB each. These file sizes represent quarterly table dumps with **every** column selected. However, BTS allows us to select which columns to export to the CSV, and we anticipate needing about **half** of the columns that are provided in each table. One of the main obstacles that these datasets might present is the fact that the two tables need to be joined together on a primary key column. The plan is for every pair of CSV files, load them into a pandas dataframe, join the dataframes, and export the joined dataframe back to a CSV file. From there, we plan on using mySQL with Google Cloud SQL to store this joined version of the two original CSVs. In terms of data processing, we need to filter out all rows that have number of passengers greater than 2. Furthermore, for datasets that date back to 2016 and further, we need to update their ticket price column to adjust for potential inflation. The goal is to bring historical prices to "today" prices.

B. Scraped Data

While not the main focus of our application, there is potential for us to scrape flight data if we are given the time. If we do find ourselves in this situation, we would have to scrape flight data in such a way that it fits in with our existing historical database. The desire for this data is derived from the "freshness" factor of live flight data that the application could benefit from.

IV. Models and Algorithms

A. Basic Algorithms/Tool and Library Overview

The majority of this application will be properly retrieving and analyzing historical data given by the Bureau of Transportation Statistics. As explained in the Historical Data section, CSV files are downloaded from the Bureau of Transportation Statistics in pairs and are combined using the pandas python library. Once combined, mySQL is used with Google Cloud SQL to store the resulting CSV. Once information is properly stored, we analyze relevant flight information for a user based on airports in their location (and if specified, within their driving distance). In order to find all airports within a radius of the user, we will use Google's GeoCode API along with its Maps API. Some of the user input that will be taken into consideration prior to any analysis is how far the user wishes to travel pre-flight, how many stops the user is willing to take during the flight, the timeframe for when the user can board, etc. At its core, our analysis will produce a table of the average ticket cost at each airport within the user's radius, along with how far away each airport is from the user. We want to also give the user a predicted "driving cost", which will initially be derived using a general, national gas price average. This average is subject to change if we find a way to dynamically determine gas prices based on location. This ordered output allows the user to see many different scenarios and hopefully help them make a decision on which airport to fly out of. As implied, most of the analysis done by the team will involve basic statistics and aggregate functions, with no initial need for any machine learning techniques, but we will see what arises as a result of our current approach.

Once our goals for the historical data analysis above are complete, we plan on transitioning to a web-app focus to add a real user interface. The plan is to use a Django backend to integrate the analysis code, which is also going to be written in python.

If time permits, there is a desire to scrape live flight data, but this is not a priority. BeautifulSoup and other relevant libraries will be used if this point is reached.

B. Models Produced

The outcome for this project will heavily depend on the graphs and tables produced to represent the manipulated data. The information presented must be in a format that allows for the average user to be able to interpret the data and make a budget decision with ease. First and foremost we want to be able to present the historical trends for the ticket that the user has selected (i.e. trend of historical ticket prices from airport X to airport Y).

Besides this preliminary graph, several tables will also be produced. These tables will focus on the direct cost comparisons between the nearest airports, average flight duration, as well as averages and other miscellaneous data that we deem would be useful to the user. Included below is an example of what a table relating to average ticket cost per airport could look like.

	Avg cost	Driving time	∼Driving cost
Airport 1	$456	1 hr	$5
Airport 2	$525	0.2 hr	$0.50
...			

TABLE I
EXAMPLE OF OUTPUT TABLE

V. TIME-LINE

A. Sprint 1

Set up the development environment for the team to use. This includes getting a Google Cloud SQL instance running and a requirements.txt document for everyone to install the dependencies locally. The goal for the Database Team is to download, clean, and insert at least **one** quarter of data into the SQL database (i.e. 2018 Quarter 1). The goal for the Data Analysis Team is to properly set up data retrieval and execute basic queries against the database.

B. Sprint 2

The goal for the Database Team is to download, clean, and insert **twenty** quarters of data (i.e. 2015-2018) into the SQL database. This might involve writing a script to do the insertions and possibly finding away to avoid manually downloading 40 CSV files. The goal for the Data Analysis Team is to write a program that returns a list of airports within a driving range of **x** miles of a user. This list should contain the average cost of a ticket from that airport to the destination airport specified by the user. The driving distance to the source airports, along with an estimated driving cost (using a general, national average of gas prices), should also be part of the list.

C. Sprint 3

The goal for the Database Team is to transition towards a Backend Team and begin work on a web application. The team should successfully deploy a Django application that queries the Google Cloud SQL database. The goal for the Data Analysis Team is to include more output for the user as described in section IV, including visualizations using a JavaScript framework (i.e. D3.js or Chart.js).

D. Sprint 4

The goal for the Backend Team is to successfully setup the apps within Django to allow for an easy transition of existing python code from the Data Analysis Team. The goal for both teams at this point is to successful integrate existing Data Analysis code into the Django application. The end result should be a web app that asks for the user input that was previously entered using the command line in previous sprints. Both teams should also work on UI to make the web app relatively appealing, although simplistic.

TEAM ROLES

The Database Team/Backend Team

- Andrey Karnauch - Team Lead, Google Cloud SQL setup and instruction, Django lead
- Rojae Johnson - Compose data merge and data insertion scripts
- Matt A. - Perform data collection and insertion

The Data Analysis Team

- Cai John - Team Lead, Visualizations, Compose queries
- Matt K. - Analyze query results and produce user output

REFERENCES

[1] Bureau of Transportation Statistics. `https://www.transtats.bts.gov/tables.asp?db_id=125&DB_Name=`