Flight Delay Rate Analysis and Prediction:

Group Project Part One

Guanzhi Wang, Boyang Wei, Xi Yang, Mengtong Zhang, Mengzhi Zhou

Abstract

This paper investigates the factors related to the delay rate of flights from top 20 large airports in the United States and makes prediction for future flights based on historical records. Specifically, we focus on gathering, cleaning and analyzing the data of flights, weather and other factors that have correlation to the flight delay rate. One of the goals of this project is to evaluate the different factors and formulate an approach to predict the delay rate for any future flights in the United States.

1. Introduction

Flights can be delayed due to multiple factors such as weather, air control and so on. Even though many of these factors have random occurrence patterns, being able to evaluate and predict how these factors affect the delay rate is meaningful not only for passengers but also for airline companies. With the increased volume of time series fight data, we begin by using Python3 to understand, clean and transform the data. We will then explore and build models to generalize and predict the delay patterns for future flight datasets.

2. Data Exploratory

2.1 Data Background

Each day, flights information has been inputted into the repository and we extract the flight status data chronologically. Raw flight data consist of more than 15 attributes including important features correlated to the prediction of delay rate such as: departure/arrival airport, weather and actual/scheduled time for flight's departure and arrival. Additionally, the weather data mapped to each specific row of flight data are retrieved via 'Metar_Ex' for more detailed information and future forecast.

In summary, there are two types of data collected: flight status data and weather data. Flight status data likes departure time and arrival time, could be used to identify whether flight is delayed or not. Except for departure time and arrival time, information about airports and aircrafts could be used to analyze their effects on delay rate Weather data are another important data source for analyzing the causation of delay rate. We primarily investigate in the factors within weather data to determine which extent will each weather factor affects delay rate. After a comprehensive analyzing of weather data, this paper will have answers for all those questions.

With the data collected, we could determine which factors influence delay rate dominantly and help passengers to predict future flights delay rate. Furthermore, collected dataset could be used to find which factors influence flight duration. Factors including flight route, weather and aircraft type might have some extent of effect in flight duration. Evaluating of these factors not only helps airports better control air-traffic but also is meaningful for passengers to stay punctual to their schedules.

2.2 Data Collection Methods

Currently, the primary method for acquiring data for flights and weather is by API. Typically, to access data via API, the 'requests' function is being utilized in Python 3. Python package 'requests' provides the ability for obtaining data from REST API and inputting data into Python by JSON format for analyzing. In this project, the source of flight status and weather data is acquired by 'FlightAware API' which provides comprehensive data attributes to analyze relationship between delay rate and other factors.

During API data collection process, we inputted a local csv file that contains top 30 airports' name into Python and passed the names of them as parameters; then we used 'departed' query method to get departed information of those 30 airports. At this point, flight identity, the required parameter in 'Flightinfo' query method, had already been included in collected dataset. By using 'Flightinfo' query method, we can generate the raw flight status dataset directly via 'FlightAware API'.

Moreover, the weather conditions of the top 30 airports are obtained by 'Metar_Ex' query method. At first, we transformed the normal date format that we need into UNIX timestamp.

Then, by adjusting the ‘HowMany’ and ‘offset’ attributes in API request, we can require 24 different weather information for the specific airport per hour. The weather dataset contains several aspects of weather such as cloud issues, air temperature, atmospheric pressure, visibility, etc.

2.3 Data Issues

After the first inspection of collected dataset, there are several data issues: missing value, redundant attributes, and incorrect data formats. In flight status dataset that collected via ‘FlightAware API’, there are several missing values in columns such as ‘aircraft type’, ‘route’ and ‘field_altitude’, ‘etc. Blank space, ‘0’, and ‘-1’ are also presented in lists. For data format, columns ‘actualdeparturetime’, ‘estimatedarrivaltime’, and ‘actualarrivaltime’ need to be transferred to correct date time format by special rules. In original dataset we collected, attribute values in those columns are a series of number with no meaning, such as: ‘1538152864’. Furthermore, column ‘ident’ also needs to be transferred to matched airline name and numbers by ‘ICAO’ flight format rule.

3.Data Cleaning

3.1 Cleaning Methods and Steps

Firstly, all the time data we obtained from 'Flightaware API' are in UNIX epoch time form, which are seconds since 1970. Thus, we need to transform UNIX epoch time back into real time form for future work. To perform this task, we use 'ctime' function in 'time' package in Python. Function 'ctime' convert a time expressed in seconds since the epoch to a string representing local time. In Fig. 3.1 below, part of our raw data is shown and an example of using 'ctime' function to decode UNIX epoch time is shown. From the result, we can see that the UNIX epoch time has been converted to normal time as a string. This string contains full date and time, so we continue separate the objects in the transferred real time. Our first trial of separating the real time string is using the space between the objects and 'split' function. However, in Fig. 3.1, there are two white spaces between month and day but between other objects, there are only one white space. If the day is two digits like 13th, the white space between month and day will change to one. Thus, we choose to use the location of characters to separate the real time string we obtained because the characters in the times are are always 24. We define two functions to clean the time data: fun_sep_time and decode_time. The first function fun_sep_time is used to separate the time and date objects we obtained into five new columns in the dataframe and delete the original raw data column. Inside fun_sep_time, we call the second function decode_time to decode raw time data and to separate objects in the decoded real time string.

filed_time	filed_departuretime
1538717168	1538916540
1538717168	1538907900
1538630802	1538830140
1538630802	1538821500
1538744627	1538743740

```
In [1]: import time

In [2]: time.ctime(1538916540)
Out[2]: 'Sun Oct  7 08:49:00 2018'
```

Figure 3.1 Clean time data by decoding UNIX epoch time.

During cleaning time data, we also observe that two of our crucial parameters, actual arrival time and actual departure time have some missing values. If one of these two parameters are missing, we cannot perform our research. Thus, we choose to delete the rows that have missing actual arrival or departure time. In Fig. 3.2, the result of visualizing and deleting missing actual arrival and departure time of one of our datasets is shown.

```
We have 904 Missing actual arrival time.

We have 825 Missing actual departure time.

We have  3865  valid rows of data in this dataset.
```

Figure 3.2 Result of missing actual arrival and departure time data during cleaning process.

Redundant data is another issue in the datasets we collected. In Fig. 3.3, part of our raw data set is shown. There are six columns related to origin and destination. The first two columns are the ICAO code of origin and destination. The third and fourth columns are the airport name and city name of the origin. The fifth and sixth columns are airport and city name of the destination. We find that the ICAO code of origin and destination record the same information with the airport name of origin and destination. Also, there are some invalid characters in some of the airport names. Thus, we choose to delete the origin and destination airport names. For origin and destination city name, we choose to keep these two columns because some cities have more than one airport.

origin	destination	originName	originCity	destinationName	destinationCity
KATL	KDTW	Hartsfield-Jackson Intl	Atlanta, GA	Detroit Metro Wayne Co	Detroit, MI
KMCO	KATL	Orlando Intl	Orlando, FL	Hartsfield-Jackson Intl	Atlanta, GA
KATL	KDTW	Hartsfield-Jackson Intl	Atlanta, GA	Detroit Metro Wayne Co	Detroit, MI
KMCO	KATL	Orlando Intl	Orlando, FL	Hartsfield-Jackson Intl	Atlanta, GA
KATL	KDTW	Hartsfield-Jackson Intl	Atlanta, GA	Detroit Metro Wayne Co	Detroit, MI
KMCO	KATL	Orlando Intl	Orlando, FL	Hartsfield-Jackson Intl	Atlanta, GA
KATL	KDTW	Hartsfield-Jackson Intl	Atlanta, GA	Detroit Metro Wayne Co	Detroit, MI
KMCO	KATL	Orlando Intl	Orlando, FL	Hartsfield-Jackson Intl	Atlanta, GA
KATL	KDTW	Hartsfield-Jackson Intl	Atlanta, GA	Detroit Metro Wayne Co	Detroit, MI
KMCO	KATL	Orlando Intl	Orlando, FL	Hartsfield-Jackson Intl	Atlanta, GA
KATL	KDTW	Hartsfield-Jackson Intl	Atlanta, GA	Detroit Metro Wayne Co	Detroit, MI

Figure 3.3 Part of one raw data set showing redundant origin and destination airport names.

3.2 Examples

In Fig. 3.4 below, we show the example of actual arrival time cleaning before and after the process. There are three conditions of actual arrival time shown in Fig. 3.4 below. The first condition is the raw UNIX epoch time data we collected from Flightaware API; the second condition is the decoded real time data from ‘ctime’ function; the second condition is the cleaned actual arrival time data, which is ready for following steps.

Figure 3.4 Example of raw time data cleaning before and after the process.

3.3 Future Generation

The first new feature we generated from our datasets is obtaining airline name from the flight ident number. The first three characters in the flight ident number are the ICAO code for airlines. Thus, we obtain a list of airline ICAO code and airline actual names and save the list into a csv file called 'airline_ICAO.csv'. In our cleaning process, we open this csv file and create a dictionary based on that. Then, we use this dictionary to obtain airline names. In Fig. 3.5 below, we can see that for first several rows, the airline ICAO code in flight ident number is 'JBU'. By using the dictionary containing airline names, we obtain the airline name as 'Jet Blue Airways'. Since some of the flight ident number contains invalid or unknown airline ICAO code, we define a function called 'fun_airline_name' to perform the task. This function will find airline names if the airline ICAO is valid. Otherwise, this function will return 'unknown'.

Figure 3.5 Obtaining airline names based on flight ident number.

The next step in our cleaning process is calculating arrival delay based on actual arrival and estimated arrival time. Also, based on the estimated departure time and actual departure time, we calculate departure delay. Two functions: 'cal_dep_delay' and 'cal_arrival_delay', are defined to perform this task. These two functions will read in the time data and then output two variables, one Boolean variable that show if there is delay, and one float variable that contains delay time in minutes. If the flight arrives on time, the delay time is returned as 0, and if the flight arrives early, the delay time is returned as a negative number. In all, we generate four features here, which are shown in the first four columns in Fig. 3.6 below.

In the last column of Fig. 3.6 below, ‘diff_flt’ represents the difference between planned fly time and actual fly time. If ‘diff_flt’ number is negative, the actual fly time is shorter than planned fly time. If ‘diff_flt’ number is positive, the actual fly time is longer than planned fly time. We generate this new feature because we notice many arrival and departure delay are all 0. Thus, we need to generate a new feature to research if there is actual delay.

arr_delay_sig	arr_delay_min	dep_delay_sig	dep_delay_min	diff_flt
FALSE	0	TRUE	7.4	-9.4
FALSE	0	TRUE	0.083333333	0.733333
FALSE	0	TRUE	10	-9.36667
FALSE	0	TRUE	0.533333333	-1.95
FALSE	0	TRUE	7.866666667	-2.33333
FALSE	0	TRUE	1.116666667	-5.85
FALSE	0	FALSE	-5.916666667	-6.18333
FALSE	0	TRUE	2.05	1.566667
FALSE	0	FALSE	-0.533333333	-2.73333
FALSE	0	FALSE	-1	-3.65
TRUE	0.283333333	TRUE	10.56666667	-4.28333
FALSE	-98.46666667	TRUE	5	-105

Figure 3.6 Calculated arrival and departure delay time in minutes, and difference between estimated fly time and actual fly time.