Big Data Analytics Final Report Template

Guy Farkash
Electrical Engineering
Columbia University
gf2373@columbia.edu

Abstract

With an average of 1.3 million business flights a day, with bad weather delays, many people are spending extra time doing nothing and, even more, getting late for an important meetings. If one could predict that a flight will be delayed, he could save time by avoiding it, using other flight at a different date or location, or postponing the meeting in advanced.

I. Introduction

Many business trips in the Unites States involves commuting between states, and for a businessman, this is a part of the job. While commuting, most people, can not work, meet people or be productive, that is why many people use flights to commute between states, which is the fastest option these days. In this project we will show a solution for minimizing the time of flight trips by predicting delays in advanced and choosing flights with the shortest delays. According to the Global Business Travel Association, in the year 2016, there were 448 million business flights in the United States, that is an average of about 1.3 million business flights a day. Also, according to the Bureau Of Transportation Statistics, about 6% of the flights gets delayed because of a bad weather. Overall, there is an average of 78 thousand business flights a day, that gets delayed because of a bad weather. For example, if we would reduce each delay by only 5 minutes, we would save about 10 months each day collectively. From the single businessman perspective, he can reduce frustrations, potentially avoid much more delays and know in advance the predicted delays for his flights. Further more, many business flights consists of multiple destinations. For multi destination trips, there are several route options were the destinations are ordered in different dates. For every different route the overall delay and cost is different. Our solution will suggest the order and the time of the flights one should take to avoid bad weather delays and also when there are several similar possibilities, it will suggest the best flight route with less cost.

II. Related Work

One work that is related to our project is a tool by a company called Tableau. Tableau is a big data company that specialize in creating tools that helps understand and visualize big data. Their tool compares US flight delays by airline and destination. This visualization tool can show the number of flights to each domestic airport along with the average departure delay times experienced by travelers in 2014, for airport of origin and a flight carrier. One can filter by airport or carrier, and see the longest delays ranking. The delays shown are the total delay, which can be consists of airline delays, air traffic delay, malfunction delay and weather delay.

Another related work was done by a team from Michigan university. They gathered data from various sources and applied advanced analysis of how weather and flight are connected. Their goal was to enable airlines to anticipate and deal with delays before they happen. That knowledge could enable airlines to adjust their schedules to account for weather patterns.

These two approaches can eventually help passengers to avoid delays. The visualization tool from Tableau can help passengers to choose the best ranked airline and and airport, and the work done by the team from Michigan university can help the airlines reduce the delays that results from the weather.

Our project also tries to reduce the amount of time a passenger spends while using flights to commute, and is doing so by using a different approach. Even when a passenger picks the best rated airline and airport, and even if airline reduces their delays, still, there is more ways to reduce delay time. There will always be days with bad weather were flights must go, and especially for the businessman with multiple destinations route, picking the best route can reduce the the total trip time even more.

III. System Overview

Our solution for avoiding flights weather delay includes three parts. The first part is processing the raw data to form our database, the second part is predicting bad weather delays for future flights and add it to the database, and the third part is forming a graph database and calculate the best route based on the predicted delays and the flights cost for a specific user input. We will describe each part in details.
In the first part we are processing raw data acquired from the Bureau Of Transportation Statistics. This data consists of 6 years of flights delays information.

This table show the attributes of the data downloaded:

	Field Name	Description
1	Year	Year
2	Month	Month
3	DayOfMonth	Day
4	OriginCityName	Origin City
5	DestCityName	Destination City
6	WeatherDelay	Weather Delay, in Minutes.

Overall, the total amount of data was 3.2GB, and includes separate file for each month. To use this data we first needed to process it, renaming cities names to match all the records in the data, removing unnecessary signs and adding a unique ID for each city.

The Flights fare information acquired from the US Department Of Transportation. This data consists of the latest average flights fare for 2017 for each quarter.

This table show the attributes of the data downloaded:

	Field Name	Description
1	city1	Origin city
2	city2	Destination city
3	fare	Average fare of all flights and airlines

This data was added to the delay data as another attribute.

The final processed data and its attributes is described in the this table:

	Field Name	Description
1	YEAR	The year → [2012, 2014, 2015, 2016, 2017]
2	MONTH	The month → integer number [1, 2, …, 11, 12]
3	DAY_OF_MONTH	The day of month → integer number [1, 2, …, 30, 31]
4	ORIGIN_CITY_ID	Origin city name → string
5	ORIGIN_CITY_NAME	Origin city ID → unique integer number
6	DEST_CITY_ID	Destination city name → string
7	DEST_CITY_NAME	Destination city ID → unique integer number
8	WEATHER_DELAY	Weather Delay, in Minutes → integer number
9	FARE	Fare in Dollars → floating point number

In the second part, to predict a delay for a certain flight in a specific date, we try to estimate the delay of this flight in the future, based on past records. To do so, we tested several machine learning predictors. We tested Naive Bayes, Random Forest, Multilayer Perceptron, One-vs-Rest and Decision Tree predictors. Among all of these predictors we got the best accuracy using the Random Forest predictor, but the accuracy was below 30%. The reason for the low accuracy is the fact that the data is too scarce for a predictor to achieve good results. In the data, the maximum amount of records, for each flight and date, is 45, and many of them can be just a few records. To solve this problem we took a different approach, and devised our own predictor that calculates the average delay in a certain way, and will be discussed in the algorithm section.

In the third part, to create the graph database we use the GrapheneDB platform, which we run inside Docker. We build the Origin Vertices file, the Destination Vertices file and the Nodes file, which we loaded to the GrapheneDB and formed the graph.

These three tables show the files attributes:

Origin Vertices

	Field Name	Description
1	ORIGIN_CITY_NAME	Origin city name → string
2	ORIGIN_CITY_ID	Origin city ID → unique integer number

Destination Vertices

	Field Name	Description
1	DEST_CITY_NAME	Destination city name → string
2	DEST_CITY_ID	Destination city ID → unique integer number

Nodes

	Field Name	Description
1	YEAR	The year → [2012, 2014, 2015, 2016, 2017]
2	MONTH	The month → integer number [1, 2, …, 11, 12]
3	DAY_OF_MONTH	The day of month → integer number [1, 2, …, 30, 31]
4	ORIGIN_CITY_ID	Origin city name → string
5	ORIGIN_CITY_NAME	Origin city ID → unique integer number
6	DEST_CITY_ID	Destination city name → string
7	DEST_CITY_NAME	Destination city ID → unique integer number
8	AVG_WEATHER_DELAY	Weather Delay, in Minutes → integer number
8	DELAY_RECORDS	Number of records → integer number
9	FARE	Fare in Dollars → floating point number

We use the graph database to extract the various information needed for the calculations of the best route. To find the best route we wrote an algorithm that goes through all the routes possibilities and recommends the best one. The algorithm will be described in details in the algorithm section.

The best route algorithm input consists of two lists and two dates. A list of cities, starting from the origin city, and fallows by the destinations cities, a list of numbers which represent the number of days the user needs to spend in each destination, starting date and ending date. The output of the algorithm is the best route describing a list of flights from one city to the other, and for each flight, its date, predicted delay and fare.

For convenience we build a user interface for choosing all the necessary inputs, showing the progress of the calculations and the output of the best route calculated.
A description of how to use the user interface is in the Software Package Description.

IV. Algorithm

Our delay predictor uses the average delay as a prediction. Because there is a small amount of reports in some flights at some dates, we are using the assumption that the weather is relatively similar in a 10 days widow (+5 days and -5 days from the flight date). With that assumption we can utilize

the information in adjacent days reports. For example, if there where many reports of weather delay in the day before to our subjected date, it can indicate that there is a good chance of delays also in the next day.
The predictor calculations are as follows:

For each flight in each date calculate (1) and then (2):

(1) the average delay from all the records from all the years and save the number of records.

(2) the average delay in a 10 days windows and save the number of records.

The result is the predicted delay for that flight and the number of records that were used to calculate the result. The number of records will be used as a confidence value of the prediction. We will demonstrate and example calculation in the next 3 tables.

The original data of a specific flight for Jan. 22nd:

YEAR	MONTH	DAY	ORIGIN CITY NAME	DEST CITY NAME	WEATHER DELAY
2012	1	22	New York, NY	Boston, MA	6
2014	1	22	New York, NY	Boston, MA	25
2014	1	22	New York, NY	Boston, MA	75
2016	1	22	New York, NY	Boston, MA	59

The average delay from the 4 records:

MONTH	DAY	ORIGIN CITY NAME	DEST CITY NAME	**AVG WEATHER DELAY**	**RECORDS**
1	22	New York, NY	Boston, MA	**41.25**	**4**

The average delay from a 10 days window :

MONTH	DAY	ORIGIN CITY NAME	DEST CITY NAME	**AVG WEATHER DELAY**	**RECORDS**
1	22	New York, NY	Boston, MA	**40.13**	**31**

In this example we can see that while the average delay stayed around 40 minutes in the last two tables, the confidence value increased significantly from 4 to 31, and that imply that this prediction should be close to the true delay.

We will show the predictor accuracy later in the experiment results section.

The best rout algorithm consists of several parts. First, the list of cities is converted to a list if IDs by searching the graph database for the cities names. Next, it finds all the routes possibilities given the cities provides, the starting date, the ending date and the days for each city.

For example, a route with 3 destinations have 6 different order possibilities as can be seen in the this table:

Start	**Date 1**	**Date 2**	**Date 3**	**End**
Ori	City 1	City 2	City 3	Ori
Ori	City 1	City 3	City 2	Ori
Ori	City 2	City 1	City 3	Ori
Ori	City 2	City 3	City 1	Ori
Ori	City 3	City 1	City 2	Ori
Ori	City 3	City 2	City 1	Ori

For each route possibility we can arrange the flights in different ways within that time period keeping the number of days for each city as the user provided.

An illustration of different timing possibilities:

Start...Date1...Date2...Date3.................**End**
Start.............Date1...Date2.....Date3....**End**
Start....Date1.........Date2...Date3...........**End**
Start....Date1....Date2...........Date3........**End**

After the algorithm had found all the different routes possibilities, it fetches the information from each flight (node) and calculates the the overall delay, the confidence for each delay and the overall fare. Then, the algorithm picks the best route which has the lowest delay with the highest confidence. A flight with 60 min delay and confidence of 30 will be better than a flight with 55 min delay and confidence of 10.
The calculation is: $60 / 30 = 2 < 5.5 = 55 / 10$.
The algorithm breaks ties in favor of the cheapest route.

While fetching data from the graph and calculating the results, the progress is presented in the user interface, in a form of percentages for each stage. The final results are presented as a recommendation for the best route, the overall delay, which flight to take and when, and delay and fare for each flight.

V. Software Package Description

Our software solution includes 6 files:

clean_build_data.py
best_route.py
graphdb_client.py
docker_load_images.sh
docker_start_graphDB.sh
docker_stop_graphDB.sh

The clean_build_data.py file is doing several things:
1) Cleans the raw data – removes records with no delay information, renames necessary cities names to match all the records and removes unnecessary signs.
2) Adds a unique ID for each city
3) Appends all the file to one database
4) Creates Origin cities vertices file
5) Creates Destination cities vertices file
6) Calculates the predicted delays and confidence values
7) Adds the fare information for each flight
6) Creates the node file

To apply this software on the raw data we need:
1) The raw delay data inside a folder named “delay_raw”.
2) Each month file of the raw delay data named “yyyy-mm”, for example “2012-03”.
3) The raw fares data inside a folder named “fares_raw”.
4) Each quarter file of the raw fare data named “yyyy-q”, for example “2017-3”.
5) Two empty folders “delay_clean”, “fares_clean”.
6) The clean_build_data.py file is in the same location as the folders.

To run the file we simply need to open a terminal in the file location and run the command:

```
$ python  clean_build_data.py
```

After the process had finished, we will have a database ready for creating the graph database.

We will mention that this process is done once, or when the data base needs to be updated.

To build the graph database we need to use the GrapheneDB. We are using it inside Docker, with the Docker images that were provided to us as a part of this course assignment. The Docker images are, gpostgres-alpine.tar, graphen.ai-graphdb-rest-ui-2.7-slim.tar and graphen.ai-indexer-0.8.tar.
To load them into Docker we need to open the terminal and run the command:

```
$ sudo sh docker_load_images.sh
```

Then we can start the GrapheneDB with the command:

```
$ sudo sh docker_start_graphDB.sh
```

To stop Docker we can use:

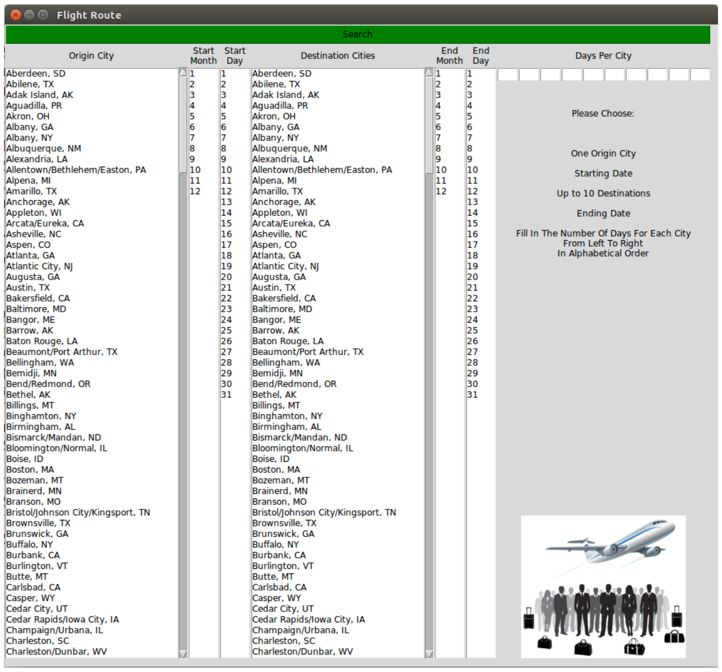
```
$ sudo sh docker_stop_graphDB.sh
```

After GrapheneDB had finished loading, we can start our best_route.py software, by running the command:

```
$ python  best_route.py
```

First, a connection is being made with Docker and then the database is loaded into GrapheneDB. This processes is using the graphdb_client.py file. When this is done, a user interface will pop up.

The User Interface



Origin City	Start Month	Start Day	Destination Cities	End Month	End Day
Aberdeen, SD	1	1	Aberdeen, SD	1	1
Abilene, TX	2	2	Abilene, TX	2	2
Adak Island, AK	3	3	Adak Island, AK	3	3
Aguadilla, PR	4	4	Aguadilla, PR	4	4
Akron, OH	5	5	Akron, OH	5	5
Albany, GA	6	6	Albany, GA	6	6
Albany, NY	7	7	Albany, NY	7	7
Albuquerque, NM	8	8	Albuquerque, NM	8	8
Alexandria, LA	9	9	Alexandria, LA	9	9
Allentown/Bethlehem/Easton, PA	10	10	Allentown/Bethlehem/Easton, PA	10	10
Alpena, MI	11	11	Alpena, MI	11	11
Amarillo, TX	12	12	Amarillo, TX	12	12
Anchorage, AK		13	Anchorage, AK		13
Appleton, WI		14	Appleton, WI		14
Arcata/Eureka, CA		15	Arcata/Eureka, CA		15
Asheville, NC		16	Asheville, NC		16
Aspen, CO		17	Aspen, CO		17
Atlanta, GA		18	Atlanta, GA		18
Atlantic City, NJ		19	Atlantic City, NJ		19
Augusta, GA		20	Augusta, GA		20
Austin, TX		21	Austin, TX		21
Bakersfield, CA		22	Bakersfield, CA		22
Baltimore, MD		23	Baltimore, MD		23
Bangor, ME		24	Bangor, ME		24
Barrow, AK		25	Barrow, AK		25
Baton Rouge, LA		26	Baton Rouge, LA		26
Beaumont/Port Arthur, TX		27	Beaumont/Port Arthur, TX		27
Bellingham, WA		28	Bellingham, WA		28
Bemidji, MN		29	Bemidji, MN		29
Bend/Redmond, OR		30	Bend/Redmond, OR		30
Bethel, AK		31	Bethel, AK		31
Billings, MT			Billings, MT		
Binghamton, NY			Binghamton, NY		
Birmingham, AL			Birmingham, AL		
Bismarck/Mandan, ND			Bismarck/Mandan, ND		
Bloomington/Normal, IL			Bloomington/Normal, IL		
Boise, ID			Boise, ID		
Boston, MA			Boston, MA		
Bozeman, MT			Bozeman, MT		
Brainerd, MN			Brainerd, MN		
Branson, MO			Branson, MO		
Bristol/Johnson City/Kingsport, TN			Bristol/Johnson City/Kingsport, TN		
Brownsville, TX			Brownsville, TX		
Brunswick, GA			Brunswick, GA		
Buffalo, NY			Buffalo, NY		
Burbank, CA			Burbank, CA		
Burlington, VT			Burlington, VT		
Butte, MT			Butte, MT		
Carlsbad, CA			Carlsbad, CA		
Casper, WY			Casper, WY		
Cedar City, UT			Cedar City, UT		
Cedar Rapids/Iowa City, IA			Cedar Rapids/Iowa City, IA		
Champaign/Urbana, IL			Champaign/Urbana, IL		
Charleston, SC			Charleston, SC		
Charleston/Dunbar, WV			Charleston/Dunbar, WV		

Please Choose:

One Origin City

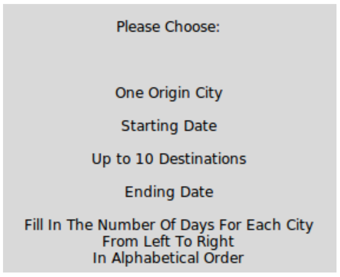
Starting Date

Up to 10 Destinations

Ending Date

Fill In The Number Of Days For Each City
From Left To Right
In Alphabetical Order

The text on the right explains how the user should choose the different entries:



Please Choose:

One Origin City

Starting Date

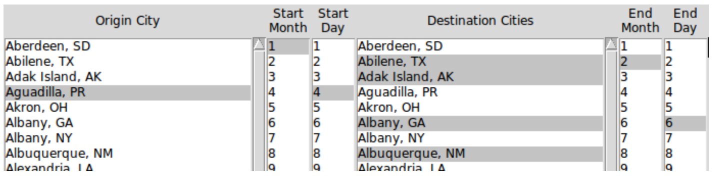
Up to 10 Destinations

Ending Date

Fill In The Number Of Days For Each City
From Left To Right
In Alphabetical Order

We chose to limit for 10 destination, from the user interface perspective, but the software as no limitation for the number of destinations.

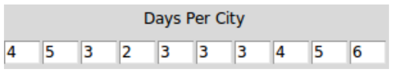
In the user interface we can see two list of cities, each one fallowed by two lists of numbers, which is the month and the day for the starting date and the end date. The first cities list allows the user to choose only one city, while the second cities list allows the user to choose several cities.



Origin City	Start Month	Start Day	Destination Cities	End Month	End Day
Aberdeen, SD	1	1	Aberdeen, SD	1	1
Abilene, TX	2	2	Abilene, TX	2	2
Adak Island, AK	3	3	Adak Island, AK	3	3
Aguadilla, PR	4	4	Aguadilla, PR	4	4
Akron, OH	5	5	Akron, OH	5	5
Albany, GA	6	6	Albany, GA	6	6
Albany, NY	7	7	Albany, NY	7	7
Albuquerque, NM	8	8	Albuquerque, NM	8	8
Alexandria, LA	9	9	Alexandria, LA	9	9

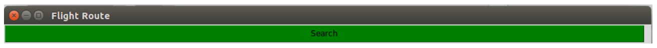
Choosing an entry is done by clicking on it with the mouse pointer. For the Destination Cities, clicking on an entry again will remove the selection.

On the top right corner there are 10 box entries for the user to input the amount of day to spend in each city. Each box correspond to a city in the order they show in the destination cities list.

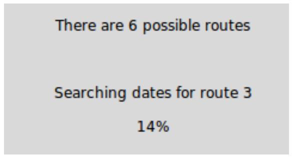

Days Per City

4	5	3	2	3	3	3	4	5	6

Clicking on the green search button on the the top, starts the best route search.



During the calculations , the progress and the final output is shown on the right:


There are 6 possible routes

Searching dates for route 3

14%

Overall, The best_route.py software is doing these things:
1) Connecting to Docker with GrapheneDB.
2) Creating a graph using the vertices and node files.
3) Creating the user interface.
4) Fetching the user input from the user interface.
5) Calculating the best route using the best route algorithm.
6) Showing the progress and the final best route result in the user interface.

VI. Experiment Results

First we will show the delay predictor test and its accuracy. We classify two kinds of errors the delay predictor can make. The prediction can be lower than the actual delay and it can also be higher than the actual delay. We will denote the lower predictions as MD (Miss Detect) and the higher predictions as FA (False Alarm). Both errors should be low but it is important that the MD prediction will be lower because MD means that the user is going to be late rather than early.

We denote these variables:
A → actual delay
P → predicted delay
MD → A < P
FA → P > A

The predictor average error is: $Error_{avg} = \frac{\sum |A-P|}{N}$

where N is the number of records

A good predictor has low $Error_{MD}$ and $Error_{FA}$, and also $Error_{MD} < Error_{FA}$ is important in our problem as stated earlier.

To test the predictor, we randomly chose a whole year of flights reports, and compare it to the predicted calculations using the other data.

The results are: $Error_{MD} \simeq 9.5\,[min]$
$Error_{FA} \simeq 20\,[min]$

The total average delay was 46 minutes.

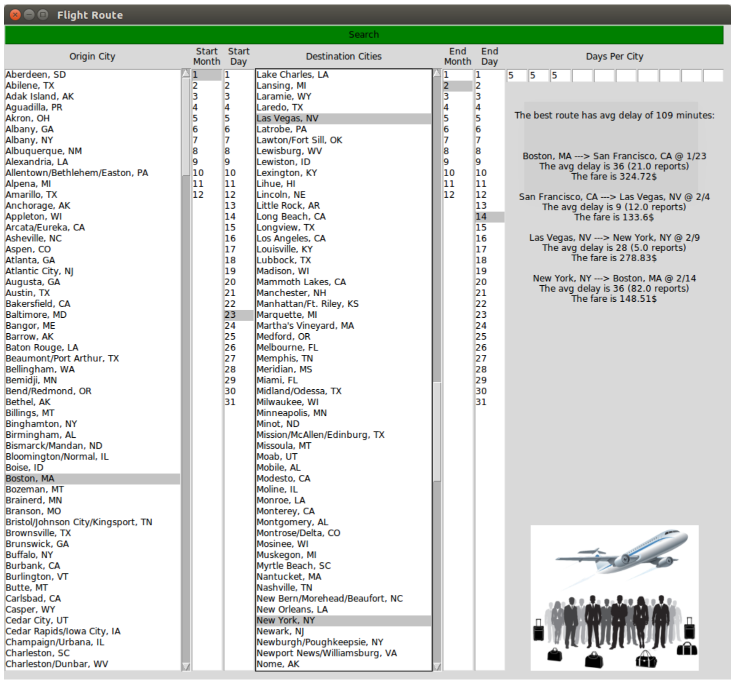
The accuracy is: $MD_{accuracy} \simeq 79\,\%$
$FA_{accuracy} \simeq 56\,\%$

The results correspond to a good predictor, and are much better than the results we got using the the ML predictors.

We will also show the output results of the best route algorithm. For example we would like to find the best route that goes through New York, Las Vegas and San Francisco, when the origin city is Boston. The starting date is Jan. 23rd and the ending date is Feb. 14th. We would like to spend 5 days in each city. The Algorithm output is:

The best route has avg delay of 109 minutes:
Boston, MA → San Francisco, CA @ 1/23
The avg delay is 36 (21.0 reports)
The fare is 324.72 $
San Francisco, CA → Las Vegas, NV @ 2/4
The avg delay is 9 (12.0 reports)
The fare is 133.67 $
Las Vegas, NV ---> New York, NY @ 2/9
The avg delay is 28 (5.0 reports)
The fare is 278.83 $
New York, NY ---> Boston, MA @ 2/14
The avg delay is 36 (82.0 reports)
The fare is 148.51 $

The user interface shows the results:



Origin City	Start Month	Start Day	Destination Cities	End Month	End Day
Aberdeen, SD	1	1	Lake Charles, LA	1	1
Abilene, TX	2	2	Lansing, MI	2	2
Adak Island, AK	3	3	Laramie, WY	3	3
Aguadilla, PR	4	4	Laredo, TX	4	4
Akron, OH	5	5	Las Vegas, NV	5	5
Albany, GA	6	6	Latrobe, PA	6	6
Albany, NY	7	7	Lawton/Fort Sill, OK	7	7
Albuquerque, NM	8	8	Lewisburg, WV	8	8
Alexandria, LA	9	9	Lewiston, ID	9	9
Allentown/Bethlehem/Easton, PA	10	10	Lexington, KY	10	10
Alpena, MI	11	11	Lihue, HI	11	11
Amarillo, TX	12	12	Lincoln, NE	12	12
Anchorage, AK		13	Little Rock, AR		13
Appleton, WI		14	Long Beach, CA		14
Arcata/Eureka, CA		15	Longview, TX		15
Asheville, NC		16	Los Angeles, CA		16
Aspen, CO		17	Louisville, KY		17
Atlanta, GA		18	Lubbock, TX		18
Atlantic City, NJ		19	Madison, WI		19
Augusta, GA		20	Mammoth Lakes, CA		20
Austin, TX		21	Manchester, NH		21
Bakersfield, CA		22	Manhattan/Ft. Riley, KS		22
Baltimore, MD		23	Marquette, MI		23
Bangor, ME		24	Martha's Vineyard, MA		24
Barrow, AK		25	Medford, OR		25
Baton Rouge, LA		26	Melbourne, FL		26
Beaumont/Port Arthur, TX		27	Memphis, TN		27
Bellingham, WA		28	Meridian, MS		28
Bemidji, MN		29	Miami, FL		29
Bend/Redmond, OR		30	Midland/Odessa, TX		30
Bethel, AK		31	Milwaukee, WI		31
Billings, MT			Minneapolis, MN		
Binghamton, NY			Minot, ND		
Birmingham, AL			Mission/McAllen/Edinburg, TX		
Bismarck/Mandan, ND			Missoula, MT		
Bloomington/Normal, IL			Moab, UT		
Boise, ID			Mobile, AL		
Boston, MA			Modesto, CA		
Bozeman, MT			Moline, IL		
Brainerd, MN			Monroe, LA		
Branson, MO			Monterey, CA		
Bristol/Johnson City/Kingsport, TN			Montgomery, AL		
Brownsville, TX			Montrose/Delta, CO		
Brunswick, GA			Mosinee, WI		
Buffalo, NY			Muskegon, MI		
Burbank, CA			Myrtle Beach, SC		
Burlington, VT			Nantucket, MA		
Butte, MT			Nashville, TN		
Carlsbad, CA			New Bern/Morehead/Beaufort, NC		
Casper, WY			New Orleans, LA		
Cedar City, UT			New York, NY		
Cedar Rapids/Iowa City, IA			Newark, NJ		
Champaign/Urbana, IL			Newburgh/Poughkeepsie, NY		
Charleston, SC			Newport News/Williamsburg, VA		
Charleston/Dunbar, WV			Nome, AK		

Days Per City

5	5	5							

The best route has avg delay of 109 minutes:

Boston, MA ---> San Francisco, CA @ 1/23
The avg delay is 36 (21.0 reports)
The fare is 324.72$

San Francisco, CA ---> Las Vegas, NV @ 2/4
The avg delay is 9 (12.0 reports)
The fare is 133.6$

Las Vegas, NV ---> New York, NY @ 2/9
The avg delay is 28 (5.0 reports)
The fare is 278.83$

New York, NY ---> Boston, MA @ 2/14
The avg delay is 36 (82.0 reports)
The fare is 148.51$

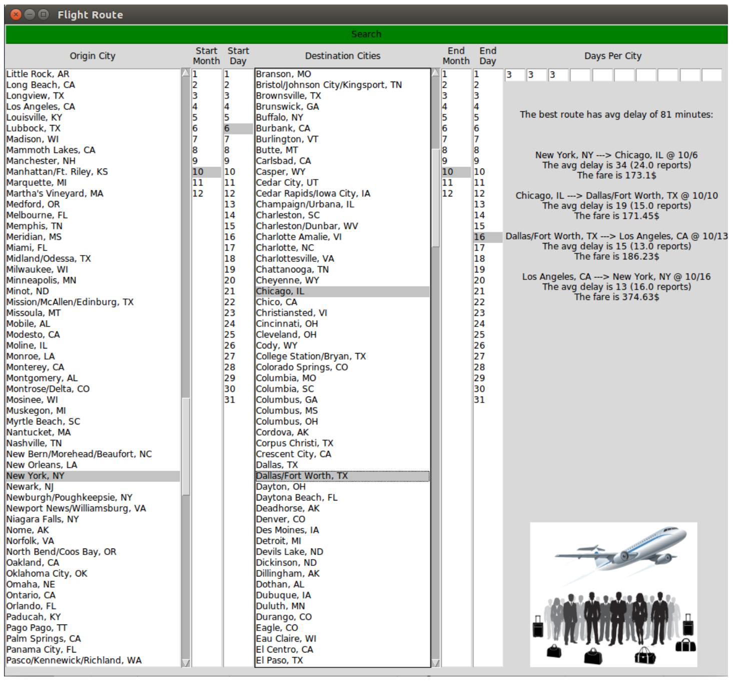
If we where to take this trip this year, the total delay of the route would have been 99 minutes, compared to our 109 minutes delay predicted. This is a FA of only 10 minutes with relatively high overall confidence value of 120 records.

If we select a different order route such as
Boston, MA → New York, NY → Las Vegas, NV → San Francisco, CA → Boston, MA
the total predicted delay is 208 minutes, which is not the best one.

We will show another example, for a route that goes through Chicago, Dallas and Los Angeles, when the origin city is New York. The starting date is Oct. 6th and the ending date is Oct. 16st. We would like to spend 3 days in each city.
The Algorithm output is:

The best route has avg delay of 81 minutes:
New York, NY → Chicago, IL @ 10/6
The avg delay is 34 (24.0 reports)
The fare is 173.1 $
Chicago, IL → Dallas/Fort Worth, TX @ 10/10
The avg delay is 19 (15.0 reports)
The fare is 171.45 $
Dallas/Fort Worth, TX ---> Los Angeles, CA @ 10/13
The avg delay is 15 (13.0 reports)
The fare is 186.23 $
Los Angeles, CA ---> New York, NY @ 10/16
The avg delay is 13 (16.0 reports)
The fare is 374.63 $

The user interface shows the results:



Origin City	Start Month	Start Day	Destination Cities	End Month	End Day
Little Rock, AR	1	1	Branson, MO	1	1
Long Beach, CA	2	2	Bristol/Johnson City/Kingsport, TN	2	2
Longview, TX	3	3	Brownsville, TX	3	3
Los Angeles, CA	4	4	Brunswick, GA	4	4
Louisville, KY	5	5	Buffalo, NY	5	5
Lubbock, TX	6	6	Burbank, CA	6	6
Madison, WI	7	7	Burlington, VT	7	7
Mammoth Lakes, CA	8	8	Butte, MT	8	8
Manchester, NH	9	9	Carlsbad, CA	9	9
Manhattan/Ft. Riley, KS	10	10	Casper, WY	10	10
Marquette, MI	11	11	Cedar City, UT	11	11
Martha's Vineyard, MA	12	12	Cedar Rapids/Iowa City, IA	12	12
Medford, OR		13	Champaign/Urbana, IL		13
Melbourne, FL		14	Charleston, SC		14
Memphis, TN		15	Charleston/Dunbar, WV		15
Meridian, MS		16	Charlotte Amalie, VI		16
Miami, FL		17	Charlotte, NC		17
Midland/Odessa, TX		18	Charlottesville, VA		18
Milwaukee, WI		19	Chattanooga, TN		19
Minneapolis, MN		20	Cheyenne, WY		20
Minot, ND		21	Chicago, IL		21
Mission/McAllen/Edinburg, TX		22	Chico, CA		22
Missoula, MT		23	Christiansted, VI		23
Mobile, AL		24	Cincinnati, OH		24
Modesto, CA		25	Cleveland, OH		25
Moline, IL		26	Cody, WY		26
Monroe, LA		27	College Station/Bryan, TX		27
Monterey, CA		28	Colorado Springs, CO		28
Montgomery, AL		29	Columbia, MO		29
Montrose/Delta, CO		30	Columbia, SC		30
Mosinee, WI		31	Columbus, GA		31
Muskegon, MI			Columbus, MS		
Myrtle Beach, SC			Columbus, OH		
Nantucket, MA			Cordova, AK		
Nashville, TN			Corpus Christi, TX		
New Bern/Morehead/Beaufort, NC			Crescent City, CA		
New Orleans, LA			Dallas, TX		
New York, NY			Dallas/Fort Worth, TX		
Newark, NJ			Dayton, OH		
Newburgh/Poughkeepsie, NY			Daytona Beach, FL		
Newport News/Williamsburg, VA			Deadhorse, AK		
Niagara Falls, NY			Denver, CO		
Nome, AK			Des Moines, IA		
Norfolk, VA			Detroit, MI		
North Bend/Coos Bay, OR			Devils Lake, ND		
Oakland, CA			Dickinson, ND		
Oklahoma City, OK			Dillingham, AK		
Omaha, NE			Dothan, AL		
Ontario, CA			Dubuque, IA		
Orlando, FL			Duluth, MN		
Paducah, KY			Durango, CO		
Pago Pago, TT			Eagle, CO		
Palm Springs, CA			Eau Claire, WI		
Panama City, FL			El Centro, CA		
Pasco/Kennewick/Richland, WA			El Paso, TX		

Days Per City: 3 | 3 | 3

```
The best route has avg delay of 81 minutes:

New York, NY ---> Chicago, IL @ 10/6
The avg delay is 34 (24.0 reports)
The fare is 173.1$

Chicago, IL ---> Dallas/Fort Worth, TX @ 10/10
The avg delay is 19 (15.0 reports)
The fare is 171.45$

Dallas/Fort Worth, TX ---> Los Angeles, CA @ 10/13
The avg delay is 15 (13.0 reports)
The fare is 186.23$

Los Angeles, CA ---> New York, NY @ 10/16
The avg delay is 13 (16.0 reports)
The fare is 374.63$
```

If we where to take this trip this year, the total delay of the route would have been 47 minutes, compared to our 81 minutes delay predicted. This is a FA of 34 minutes with relatively low overall confidence value of 68 records.

If we select a different order route such as
New York, NY → Dallas/Fort Worth, TX → Los Angeles, CA → Chicago, IL → New York, NY
the total predicted delay is 129 minutes, which is not the best one.

VII. Conclusion

In this project we build a software that finds the best flight route based on a predicted weather delay. We know that even today, predicting weather is no easy task, but our solution predicts weather delay and not the weather it self. Also, our predictions are for average delay, and we tried to minimize the Miss Detect predictions which are worse than False alarm predictions. It is better to predict higher delay when the actual delay is less and it is worse to predict lower delay when the actual delay is higher. In our testing we showed a relative good accuracy of 79% for Miss Detect. The user interface we build, is simple and convenient for every user. This tool can already help users and recommend them to choose a flight route with less delays, but there are ways to improve it as part of a future work. For instance, we can improve the predictions of delays when the flight we are looking for is in the near future, by fetching on-line reports of weather delays. We could also improve the predictions by using data from several sources, such as meteorologic weather predictions and come up with an algorithm to extract weather delays. We can also get on-line flights costs from airlines websites to get the most updated fares.
From the user perspective, we can add more options, for example, allowing the user to prefer low cost route rather than the one with less delays. Overall, our tool provides the user with the information he needs for the best route, based on the predicted delay, confidence level and fare.

References

[1] Business Travel Statistics: 23 Speedy Facts to Know
https://www.creditdonkey.com/business-travel-statistics.html#distance

[2] How big data could reduce weather-related flight delays
http://www.ns.umich.edu/new/multimedia/videos/22592-how-big-data-could-reduce-weather-related-flight-delays

[3] Compare US flight delays by airline and destination
https://www.tableau.com/solutions/workbook/big-data-more-common-now-ever

[4] Bureau Of Transportation Statistics
https://www.transtats.bts.gov/Tables.asp?DB_ID=120&DB_Name=Airline%20On-Time%20Performance%20Data&DB_Short_Name=On-Time

[5] U.S. Department of Transportation
https://data.transportation.gov/Aviation/Consumer-Airfare-Report-Table-1-Top-1-000-Contiguo/4f3n-jbg2