PROGRAMMING FOR DATA SCIENCE (ST2195)

COURSEWORK

UNIVERSITY OF LONDON

NAME: MICHELLE ANGELINE

STUDENT NUMBER: 200618238

Table of Contents

I. Introduction

A. Purpose

This report will analyze the flight data provided in Harvard Dataverse. The purpose of this report is to explain how to use R and Python to analyze the dataset to produce important insights and facts that can be used by airline companies to know their audience better.

B. Data

We will use two years of flight data from 2004 until 2005 to answer the five questions given. There are five files to be used which are 2004.csv, 2005.csv, airports.csv, carriers.csv, and plane-data.csv.

C. Method

We will use R Markdown and Jupyter Notebook to answer the questions in R using dplyr and Python using pandas respectively. For data visualization, we will use the ggplot2 package from R and matplotlib and seaborn packages from Python.

II. Answer Questions

Before answering the questions below, we set our working directory and loaded the R and Python libraries needed. From Harvard Dataset, we will use data from January 2004 until December 2005. Then, we created six tables:

1. “airports” table that consists of data on several airports in the USA.
2. “carriers” table that consists of data on USA carriers.
3. “planes” table that consists of data on different types of planes.
4. “flight2004” table that consists of data on commercial flights in 2004.
5. “flight2005” table that consists of data on commercial flights in 2005.
6. “flights” table that consists of data on commercial flights in 2004 and 2005.

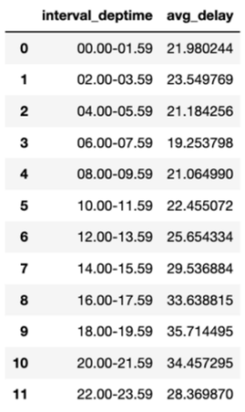
Then we filter the “flights” table to only consist of flights that are not cancelled and diverted using the filter function because our main focus is to analyze flights that departed and not diverted. We also drop duplicate rows using a distinct function in R and “drop_duplicates” in Python. In the discussion below, we do not provide all the tables and visualizations from R and Python as they are the exact same. However, all tables and visualizations can be accessed through the R Markdown and Python files.

A. When is the best time of day, day of the week, and time of year to fly to minimize delays?

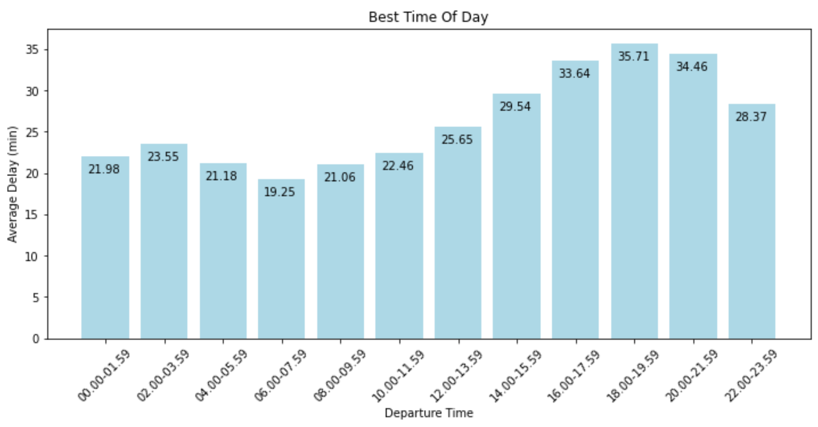
We will answer this question in three parts which are the best time of day, best day of week, and best time of year. We use “ArrDelay” as a delay measure since people tend to be concerned about what time they will arrive rather than what time they will depart.

In the best time of day, we will divide the time in 2 hours interval from the “flights” table using mutate function and named it “interval_deptime” since it is more make sense to conclude the answer in a time interval than in a particular time. Then we filter where “ArrDelay” is larger than 0 because we want to find the minimum delay. Next, we group the data by “interval_deptime” and aggregate “avg_delay” which is the average of “ArrDelay”. Finally, we arrange “interval_deptime” from ascending.

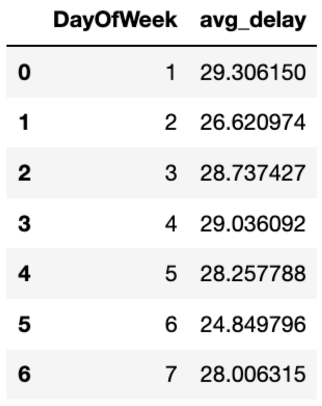
On the best day of the week, from “flights” table we filter “ArrDelay” that is larger than 0 and group the data by “DayOfWeek” and aggregate “avg_delay” which is the average of “ArrDelay”. Last, we arrange the data by “DayOfWeek”. In the best time of year, we did the exact same method, but the only difference is we group the data by “Month” and arrange “Month” from ascending to find the best time of year to fly. Then we plot the graphs using the ggplot2 package from R and matplotlib package from Python to visualize the best time of day, day of the week, and time of year to fly to minimize delays.



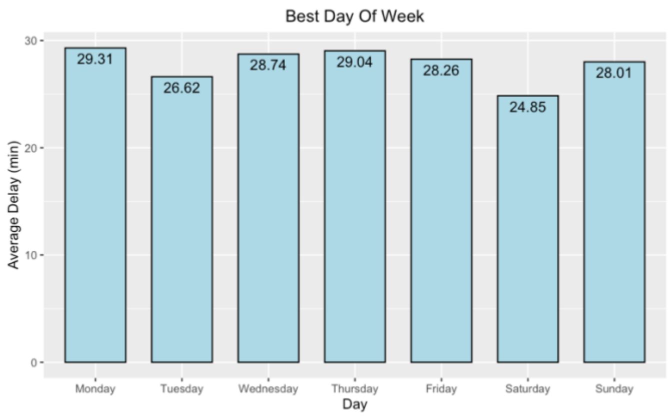
	interval_deptime	avg_delay
0	00.00-01.59	21.980244
1	02.00-03.59	23.549769
2	04.00-05.59	21.184256
3	06.00-07.59	19.253798
4	08.00-09.59	21.064990
5	10.00-11.59	22.455072
6	12.00-13.59	25.654334
7	14.00-15.59	29.536884
8	16.00-17.59	33.638815
9	18.00-19.59	35.714495
10	20.00-21.59	34.457295
11	22.00-23.59	28.369870



In the bar chart above, we found that the best time of day to travel is between 06.00 AM - 07.59 AM with an average delay of 19.25 minutes while the highest average delay is between 18.00 PM - 19.59 PM with an average of 35.71 minutes. Overall, before noon has a lower average delay than in the afternoon.

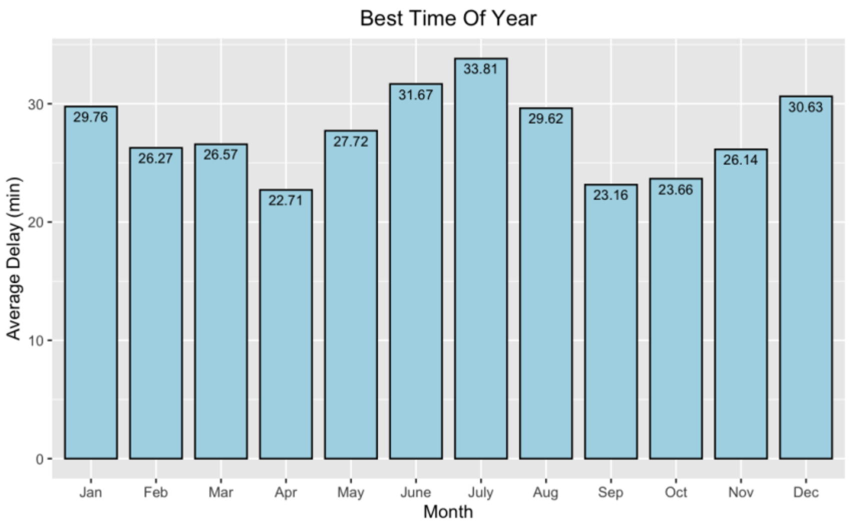


	DayOfWeek	avg_delay
0	1	29.306150
1	2	26.620974
2	3	28.737427
3	4	29.036092
4	5	28.257788
5	6	24.849796
6	7	28.006315



While there is no extreme difference in average delay between days, the bar chart shows that the best day of the week to fly is on Saturday with an average delay of 24.85 minutes, followed by Tuesday with an average delay of 26.62 minutes. We also analyze those other days except Tuesday and Saturday, the average delay is already above 28 minutes.

	Month	avg_delay
0	1	29.763226
1	2	26.270685
2	3	26.573333
3	4	22.714379
4	5	27.716673
5	6	31.670805
6	7	33.812962
7	8	29.622523
8	9	23.157299
9	10	23.663587
10	11	26.144535
11	12	30.625980



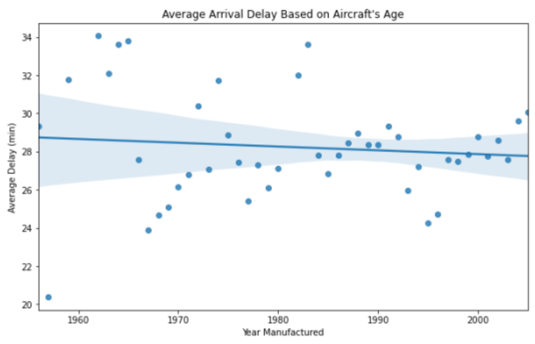
The best time of year to fly from the bar chart produced is in April with an average delay of 22.71 minutes, followed by September and October with an average delay of 23.16 and 23.66 minutes respectively. While the highest average delay is in July with an average delay of 33.81 minutes, followed by June and December with an average delay of 31.67 and 30.63 respectively. This can be caused that June, July, and December are holiday seasons which makes holiday travel chaotic. Especially in June and July since it is the summer months.

B. Do older planes suffer more delays?

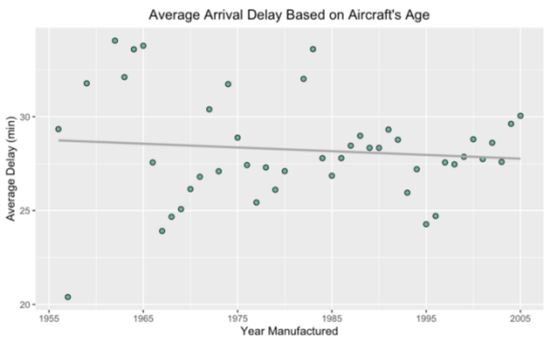
First, we have to inner join the “flights” table and “planes” table on “TailNum” and named the combined table as “planes_age” to get the year of planes manufactured and delay data in the same table. Then, we filter “ArrDelay” greater than 0 to make sure we use the data that only suffer delays. Next, we group the data by variable “year” from the “planes” table which is the year of planes manufactured. To find the delays by each year of planes manufactured, we aggregate “ArrDelay” by mean and count and we named it “avg_delay” and “total_flights” respectively. Then we arrange the data by “year” from ascending. Finally, we filter NA and column “year” which has no value, “None”, “0000”, and less than or equal to 2005 since we only observe the data until the year of 2005.

	year	avg_delay	total_flights
1	1956	29.337607	234
2	1957	20.392562	968
3	1959	31.777202	1930
4	1962	34.050831	1023
5	1963	32.108571	1400
6	1964	33.588979	1107
7	1965	33.778633	1039
8	1966	27.563073	1744
9	1967	23.910976	19343
10	1968	24.668412	32465
11	1969	25.076101	15532
12	1970	26.145650	2609
13	1971	26.799827	1159
14	1972	30.393443	244
15	1973	27.093301	3762
16	1974	31.733719	1382
17	1975	28.883915	9605
18	1976	27.422845	11924
19	1977	25.431431	6745
20	1978	27.297770	9729
21	1979	26.108193	15842
22	1980	27.099645	10146
23	1982	32.013556	2213
24	1983	33.599116	10634
25	1984	27.790122	31485
26	1985	26.856286	107199
27	1986	27.792206	91894
28	1987	28.456811	136783
29	1988	28.981263	164272
30	1989	28.340847	133218
31	1990	28.338743	146412
32	1991	29.315515	149134
33	1992	28.772336	142719
34	1993	25.951725	102372
35	1994	27.203356	133805
36	1995	24.273398	123684
37	1996	24.710179	138989
38	1997	27.562370	133871
39	1998	27.462858	277396
40	1999	27.864680	309436
41	2000	28.797712	333006
42	2001	27.741579	417188
43	2002	28.611319	350416
44	2003	27.589468	297809
45	2004	29.619524	169774
46	2005	30.048086	53404

From the query, we plot the data in scatter plots using the ggplot2 package from R and the seaborn package from Python. We set x axis as year manufactured and y axis as average delay in minutes. In the line chart below, the slope is downward sloping indicating that old planes suffer more delays.

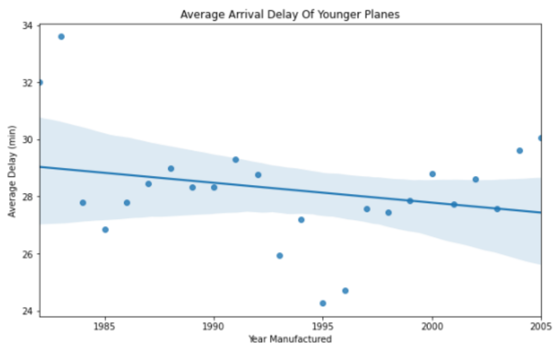


Python Visualization

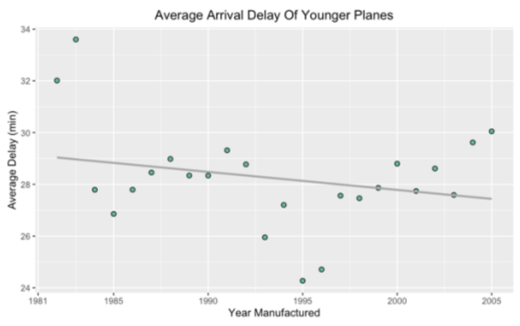


R Visualization

Then we divide the data into two parts which are from 1956 to 1980 and 1981 to 2005. We found that total flights from 1956 to 1980 are 149,932 which is very small compared to 1981 to 2005 which is 3,957,216. Since the sample size for older planes is relatively low compared to younger planes, the estimation would be less accurate. Hence, we plot the younger planes with a larger sample size to provide more accurate results. Here, we found that the slope is more steep and downward sloping. Thus, we can conclude older planes suffer more delays than younger planes.



Python Visualization

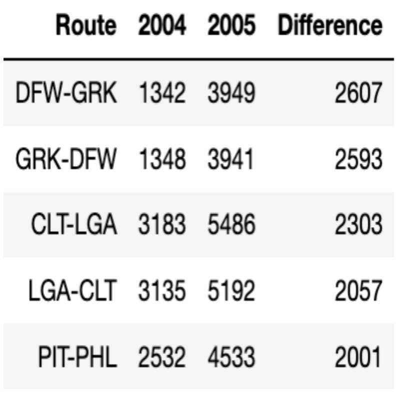


R Visualization

C. How does the number of people flying between different locations change over time?

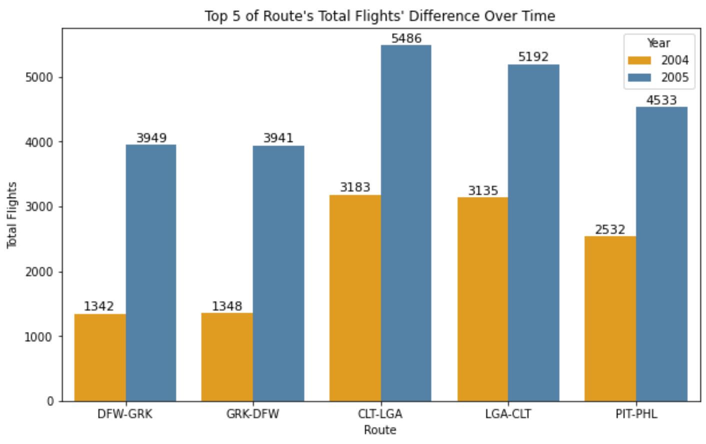
In this question, we do not know the number of people flying in each flight, so we assume it is equal. From the "flights" table, we make a new column named "Route" which concatenates "Origin" and "Dest". Then we filter the rows which contain Year equals to "2004" using the filter function and group by and count by "Route", we named the table "route_2004". Then, we rename the count column as "2004". Next, we filter again the rows which contain "Year" equals to "2005" and group by and count by "Route". Then we named the table "route_2005". Again, we rename the count column as "2005". The next step is we inner join "route_2004" and "route_2005" on "Route" to only show routes which are from both tables so we can compare total flights between 2004 and 2005, we named it as "route_merge". Then, in the “route_merge” table, we make a new column “Difference” that subtracts total flights in 2004 from total flights in 2005. Finally, we arrange the “Difference” and take the top 5 and bottom 5 and named them “top_route_merge” and “bottom_route_merge” respectively. The reason to take the top 5 and bottom 5 is to find the highest traffic and the lowest traffic to compare between years. The last step is we only select “Route”, “2004”, and “2005” columns and melt these two tables to ease the visualization process.

We use the ggplot2 package from R and the seaborn package from Python to visualize how total flights between different locations change over time. We plot the data in grouped bar charts.



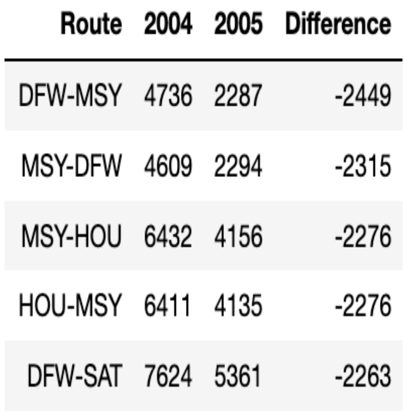
Route	2004	2005	Difference
DFW-GRK	1342	3949	2607
GRK-DFW	1348	3941	2593
CLT-LGA	3183	5486	2303
LGA-CLT	3135	5192	2057
PIT-PHL	2532	4533	2001

Top 5 Table



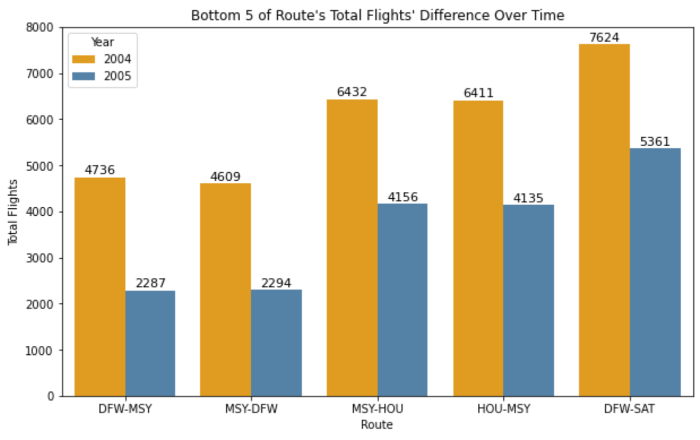
Top 5 Visualization

From the grouped bar chart above, we can see the top 5 flight routes with the highest differences (2005-2004) are DFW-GRK, GRK-DFW, CLT-LGA, LGA-CLT, and PIT-PHL respectively. In 2005, flights from Dallas/Fort Worth Airport (DFW) to Killeen-Fort Hood Regional Airport (GRK) had increased by 2,607 flights, the highest increase from other flight routes. On the opposite, flights from Killeen-Fort Hood Regional Airport (GRK) to Dallas/Fort Worth Airport (DFW) were the second-highest increasing by 2,593 total flights. So, we can conclude there were 14 more flights flying from Dallas/Fort Worth Airport (DFW) to Killeen-Fort Hood Regional Airport (GRK) rather than from Killeen-Fort Hood Regional Airport (GRK) to Dallas/Fort Worth Airport (DFW). In Charlotte Douglas International Airport (CLT) to LaGuardia Airport (LGA) route, the difference is 2,303 flights higher in 2005, while the opposite is increased by 2,057 flights. This means there are 246 more flights to LaGuardia Airport (LGA) than to Charlotte Douglas International Airport (CLT). For the PIT-PHL route, the total flights’ difference is 2,001 flights higher in 2005 than in 2004.



Route	2004	2005	Difference
DFW-MSY	4736	2287	-2449
MSY-DFW	4609	2294	-2315
MSY-HOU	6432	4156	-2276
HOU-MSY	6411	4135	-2276
DFW-SAT	7624	5361	-2263

Bottom 5 Table

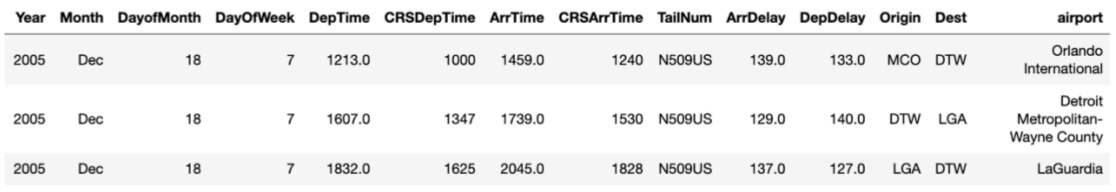


Bottom 5 Visualization

The bottom 5 flight routes with the lowest differences (2005-2004) are DFW-MSY, MSY-DFW, MSY-HOU, HOU-MSY, and DFW-SAT. In 2005, flights from Dallas/Fort Worth Airport (DFW) to Louis Armstrong New Orleans International Airport (MSY) had decreased by 2,449 flights, the highest decrease. On the opposite route (MSY-DFW) the total flights had decreased by 2,315 flights. The same thing happened to MSY-HOU and HOU-MSY had decreased by 2,276 flights for both routes. The last one in the observation is the DFW-SAT route which decreased by 2,263 flights in 2005.

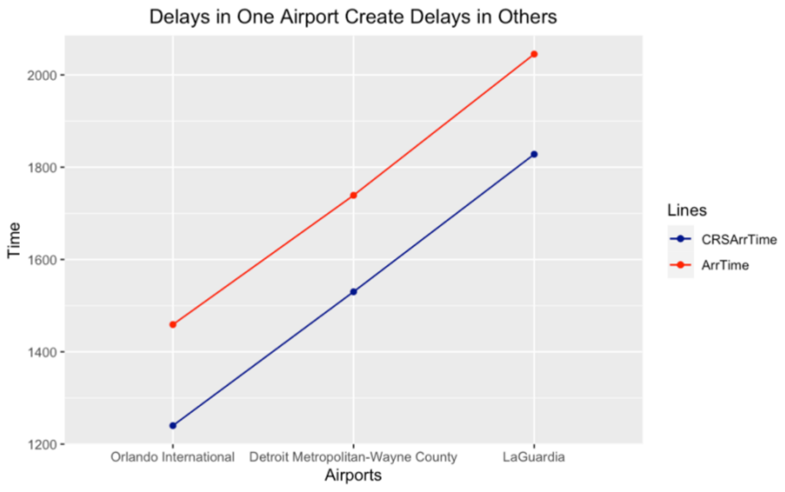
D. Can you detect cascading failures as delays in one airport create delays in others?

First, we inner join the "flights" table and "airports" table by "Origin" from the "flights" table equals to "iata" from the "airports" table. Then we select several attributes which are "Year", "Month", "DayofMonth", "DayOfWeek", "DepTime", "CRSDepTime", "ArrTime", "CRSArrTime", "TailNum", "ArrDelay", "DepDelay", "Origin", "Dest", "airport" and called the table as "cascadingfailures". From the "cascadingfailures" table we filter "DepDelay" and "ArrDelay" greater than 100. Then from the first filter, we pick a specific Month which is 12. Next, we observe the table after the first two filters and pick specific "DayofMonth" and "TailNum" which are 18 and N509US respectively. Finally, we got the final table that helps to detect if delays in one airport create delays in others.



Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	TailNum	ArrDelay	DepDelay	Origin	Dest	airport
2005	Dec	18	7	1213.0	1000	1459.0	1240	N509US	139.0	133.0	MCO	DTW	Orlando International
2005	Dec	18	7	1607.0	1347	1739.0	1530	N509US	129.0	140.0	DTW	LGA	Detroit Metropolitan-Wayne County
2005	Dec	18	7	1832.0	1625	2045.0	1828	N509US	137.0	127.0	LGA	DTW	LaGuardia

From the table above, we can see that plane with a tail number of N509US flew from Orlando International Airport late. The scheduled departure time is at 10.00 AM and the scheduled arrival time is at **12.40 PM**. However, the actual departure time is at 12.13 PM and the actual arrival time is at **14.59 PM** in Detroit Metropolitan-Wayne Country Airport. This will result in a delay in another airport as the next flight of this plane is scheduled at 13.47 PM but cannot fly from Detroit Metropolitan-Wayne Country Airport on time due to the previous flight delay. Thus, the second flight flew at 16.07 PM and arrived at **17.39 PM** while it was supposed to arrive at **15.30 PM** at LaGuardia Airport. This will impact the third flight of this plane too. Thus, we can conclude that delays in one airport cause delays in others.



In the graph above, we see that the scheduled arrival time (CRSArrTime) line is below the arrival time (ArrTime) line. This shows that the arrival time is delayed and not as scheduled. Thus, this issue will impact the next scheduled flight in other airports and eventually create another delay.

E. Use the available variables to construct a model that predicts delays

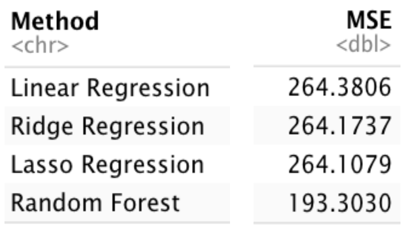
To construct several models that predict delays, we will skim the “flights” table and use nine numerical values which are “Year”, “Month”, “DayofMonth”, “CRSDepTime”, “CRSArrTime”, “DepDelay”, “ArrDelay”, “FlightNum”, and “Distance”. Since we used only numerical variables, we can construct several models such as Linear Regression, Ridge Regression, Lasso Regression, and Random Forest. First, there are no missing values from the table so we can directly set seed and take a sample of 100,000 rows from the “flights” table and named the sample table “sample_flights” because if we use all data, there will be millions of rows that are too heavy to use in the models.

In R, we set up a task that will be doing regression on the “sample_flights” dataset with “ArrDelay” as a target variable. Then, we set seed to ensure we get the same result for randomization and split the train and test data as 0.7 and 0.3 respectively. Finally, we set a learner “regr.lm” for linear regression from the “glmnet” package and train the data and extract the predictions for the test data. In ridge regression and lasso regression, we set a learner “regr.glmnet” and differentiate the models by using alpha = 0 and alpha = 1 for ridge regression and lasso regression respectively. Then train the data and use the predictions for the test data again. The same method is repeated for the random forest model, the only difference is we set a learner "regr.ranger". Finally, we find the Mean Squared Error (MSE) from each models.

In Python, we use “Year”, “Month”, “DayofMonth”, "CRSDepTime, "CRSArrTime", "DepDelay", “FlightNum”, and “Distance” from the “sample_flights” table as the explanatory variables and “ArrDelay” as the response variable. We named the explanatory variables as “train_X” and response variable as “train_Y”. Then we split the train and test data as 0.7 and 0.3 respectively.

Next, we create four learners such as “LinearRegression()”, “linear_model.Ridge()”, “linear_model.Lasso()”, and “RandomForestRegressor()” and named them “linear”, “ridge”, “lasso”, and “random_forest” respectively. Then we fit the learners to “train_x” and “train_y” from the train set. Finally, we predict “train_x” and “test_x” from each learner and find the Mean Squared Error (MSE) and R Squared.

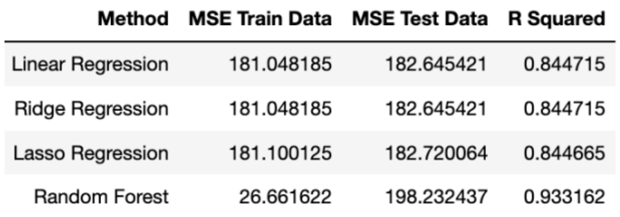
From the R table, the Random Forest model shows the lowest MSE at 193.30 while the other models have an MSE of around 260. Thus, we can conclude that Random Forest is the best method to predict delays. However, the random forest model has a drawback which tends to overfit the model, so we need to use hyperparameter tuning which requires an advanced device since the data we used consists of millions of rows.



Method <chr>	MSE <dbl>
Linear Regression	264.3806
Ridge Regression	264.1737
Lasso Regression	264.1079
Random Forest	193.3030

R Table

From the Python table, we found that linear, ridge, and lasso regression have Mean Squared Error (MSE) of around 182 in the test data and R Squared around 0.845. We also notice that linear regression and ridge regression have the same results. For random forest, it has low MSE in train data and high MSE in the test data, the gap between them is so far. This might be due to the overfitting model. Hence, we can reduce the overfitting by using hyperparameter tuning and increasing the sample size.



Method	MSE Train Data	MSE Test Data	R Squared
Linear Regression	181.048185	182.645421	0.844715
Ridge Regression	181.048185	182.645421	0.844715
Lasso Regression	181.100125	182.720064	0.844665
Random Forest	26.661622	198.232437	0.933162

Python Table

III. References

Cridland, J. C. (n.d.). *How to choose a sample size*. Tools4dev. Retrieved March 21, 2022, from https://tools4dev.org/resources/how-to-choose-a-sample-size/

Where developers learn, share, & build careers. Stack Overflow. (n.d.). Retrieved March 25, 2022, from https://stackoverflow.com/

ST2195 Study Guide