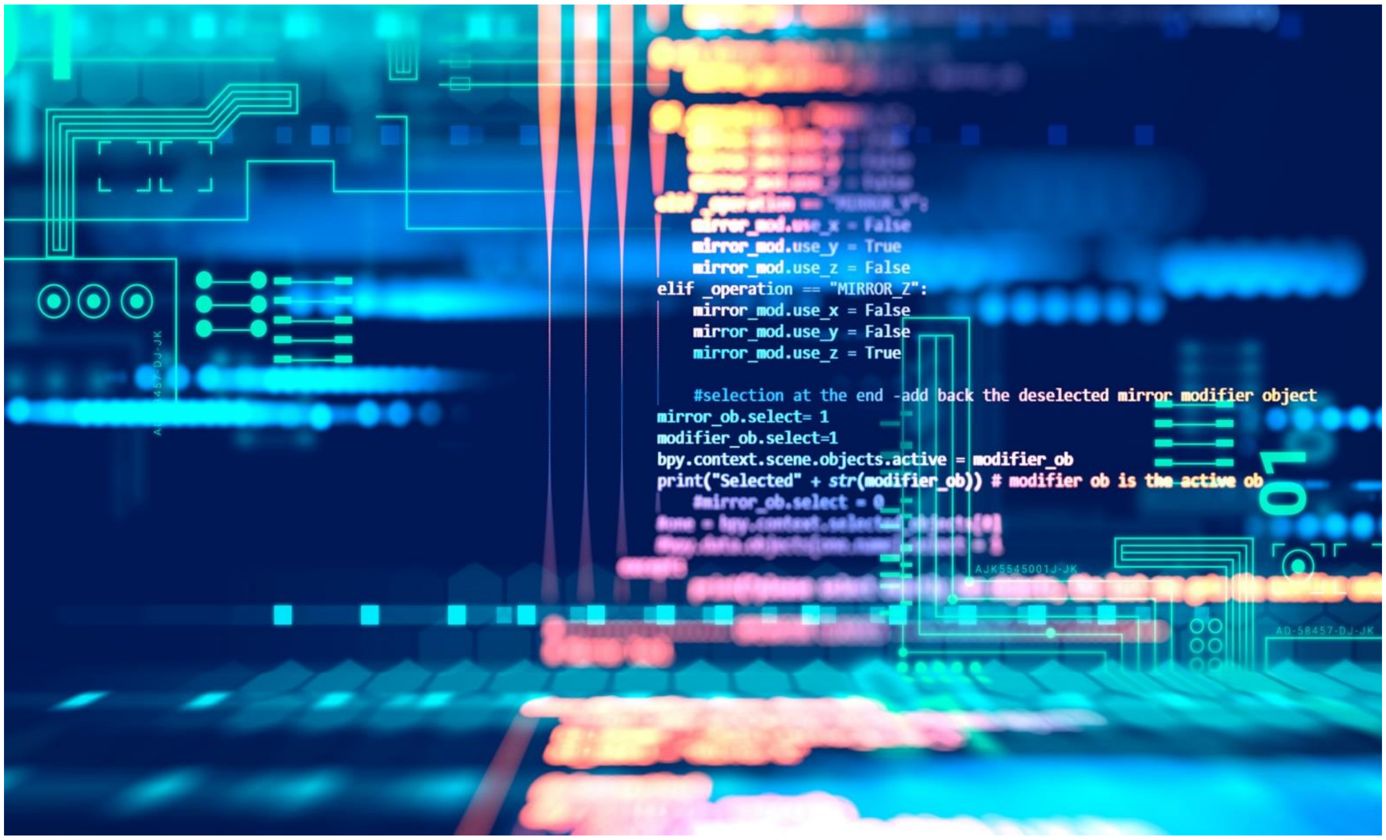
Coursework: Programming for Data Science – ST2195



Name: Binari Yashodara Wimalaratne

UoL Student Number: 200644307

Table of Contents

Introduction

This report discusses the findings from the datasets provided by the 2009 ASA Statistical Computing and Graphics Data Expo, with information on all flight arrival and departures by different carriers within the USA. The five questions are answered using Python and R files, where deep analysis on each question was carried out.

Data Cleaning

First, the two datasets for 2006 and 2007 were imported and combined together. The hours and minutes were then extracted for each "CRSDepTime" and "CRSArrTime" (scheduled departure and arrival times) by taking the first and last two characters of each string. Once this was done, the two variables were changed to date-time format. Finally, all null values were removed from the dataset.

Question 1

When is the best time of day, day of the week, and time of year to fly to minimise delays?

To evaluate the best time of day, the hours were first extracted from the "CRSDepTime" variable (scheduled departure time) by taking out the first two characters in each time. Then, a new column was created in the main dataset called "Hour" and a table was created to compare the hour of day with the average departure delay. A bar graph was then created using the table, as seen below. From these graphs, it can be seen that approximately 5:00am is the best time of day to reduce departure delays.

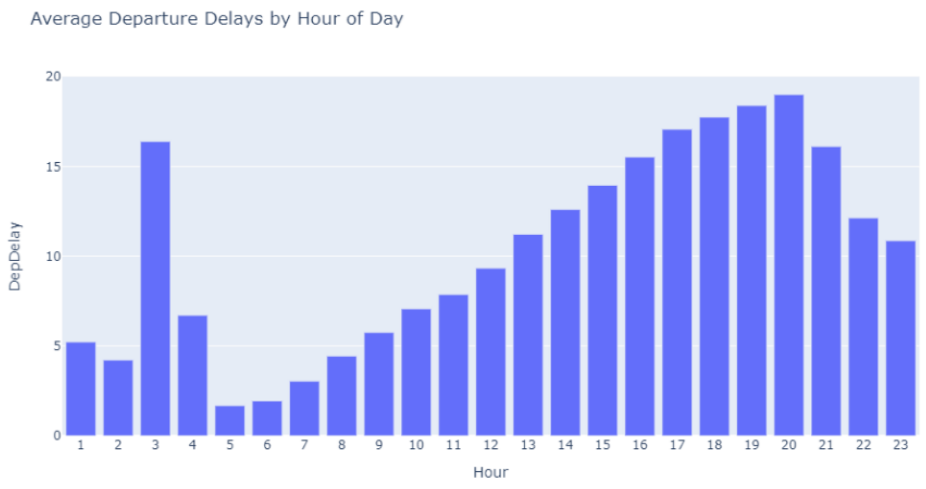


Figure 1: Hours of Day graph in Python

For the best day of week and time of year, average arrival delay was taken as the variable to minimise delays on.

To find the best day of week, a variable was mapped to match the weekdays to the numbers 1-7 (1 being Monday). Then, a new column was created named "Weekday". Another table was then made to compare the weekday with average arrival delay. A bar graph was then plotted for this data as well. From the Figures 1 and 2 below, it can be seen that Saturday is the day which minimises arrival delay.

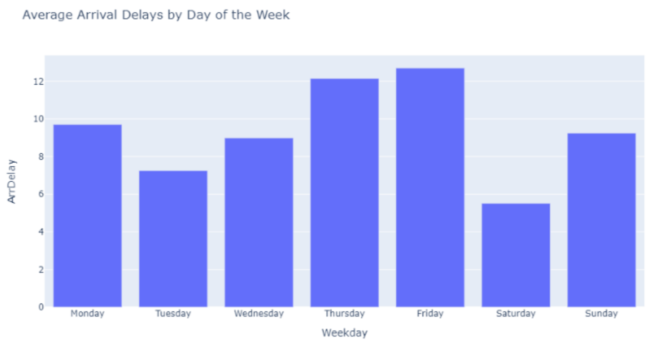


Figure 2: Weekday graph in Python

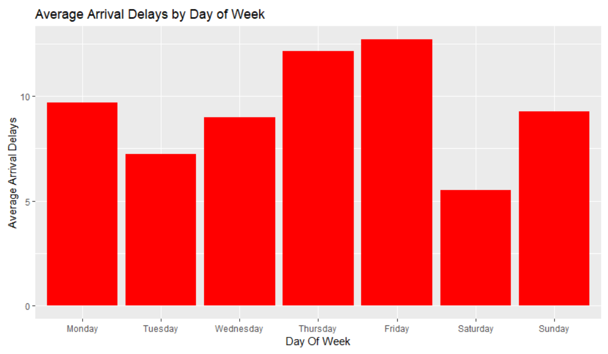


Figure 3: Weekday graph in R

A similar procedure was done to find the best time of year, analysing average Arrival Delay by Month. The bar graphs (Figures 4 and 5) plotted for this show that November is the month where the lowest arrival delays are experience within these two years.

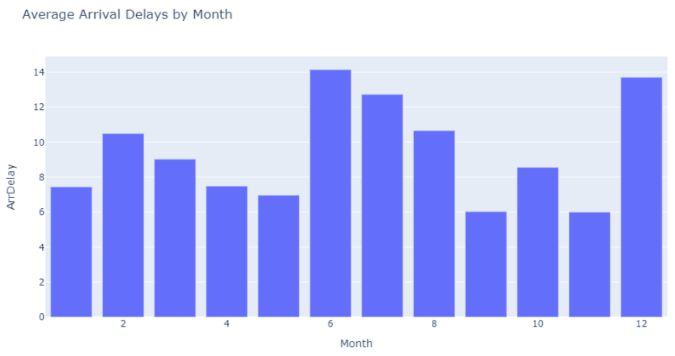


Figure 4: Month graph in Python

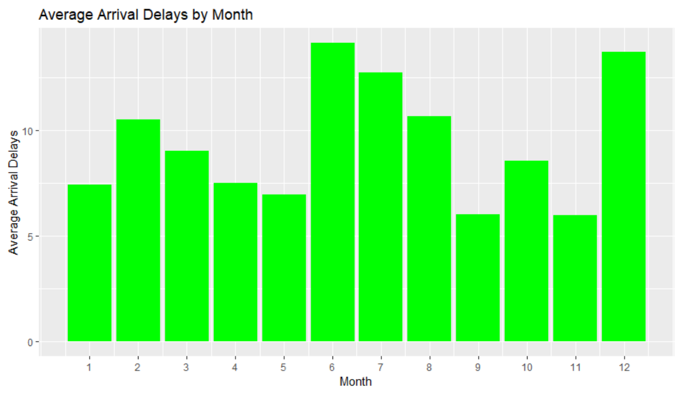


Figure 5: Month graph in R

Question 2

Do older planes suffer more delays?

Different procedures were taken for Python and R, as R repeatedly returned memory errors and was unable to merge the “plane-data” dataset with the main dataset. Therefore, both procedures are explained separately.

To analyse this question in Python, first the plane-data dataset was imported and all null values were removed. Then, a subset was made with the variables which were needed; “talinum”, “issue_date”, and “year”. Then, the variable names were changed. “talinum” was renamed to “TailNum”, as the name must match the column from the main dataset in order to merge the two datasets together. The other two variable names were also changed to make them a bit more direct. Then, this subset was merged with the main dataset to make a new data frame.

Then, the departure and arrival delays were filtered so that only the positive delays were taken, as negative delays would indicate early departures and arrivals, which are not relevant for this question. Then, to find the total delay of a flight, the departure and arrival delay of each flight were added together, and a new column was created using this.

Next, a table was made grouping plane manufacture year (renamed “year” variable) with departure delay, as well as total delay. Departure delay was observed, as this would be useful to compare delays between different years of manufactured planes. Then, two scatter plots were made to examine the relationship between plane manufacture year and departure/total delay.

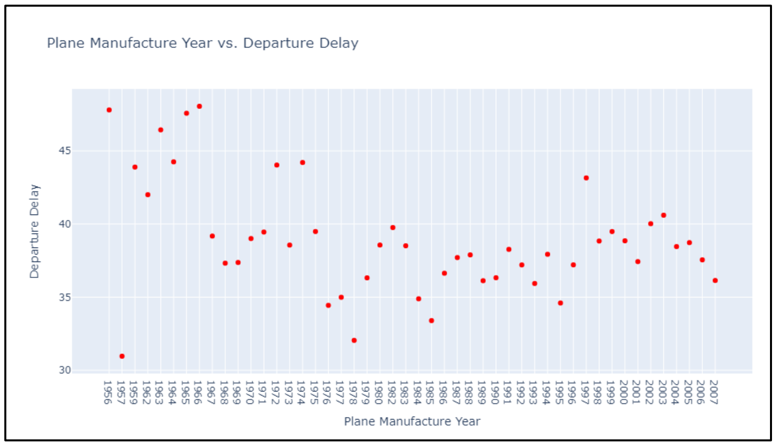


Figure 6: Departure Delay in Python

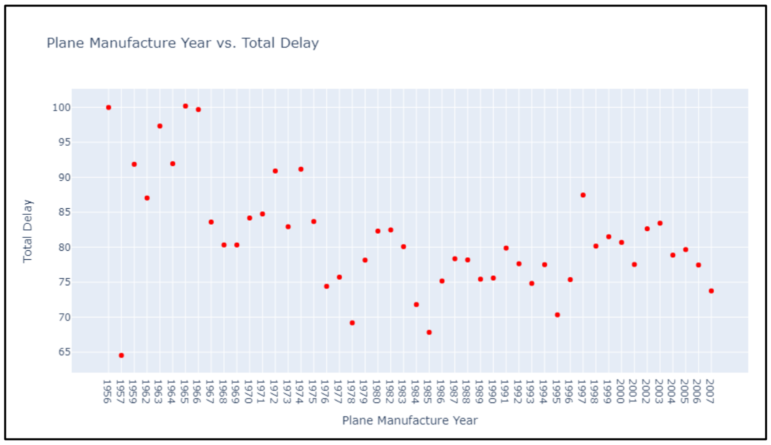


Figure 7: Total Delay in Python

In R, the procedure differed slightly because the plane-data could not merge with the main dataset, as the size was too large. Therefore, a random sample of the main dataset was taken (5,000,000 values). Then, to remove the empty rows from the plane-data dataset, a function was used to replace all blank rows with “NA”, and then they were filtered out and removed. Then, the sample of the main dataset was merged with the cleaned plane-data dataset. Similar to in Python, the departure and arrival delays were filtered to keep only the positive delays, and a total delay variable was created again.

Then, the two scatter plots were made to examine the relationship between plane manufacture year and departure/total delay.

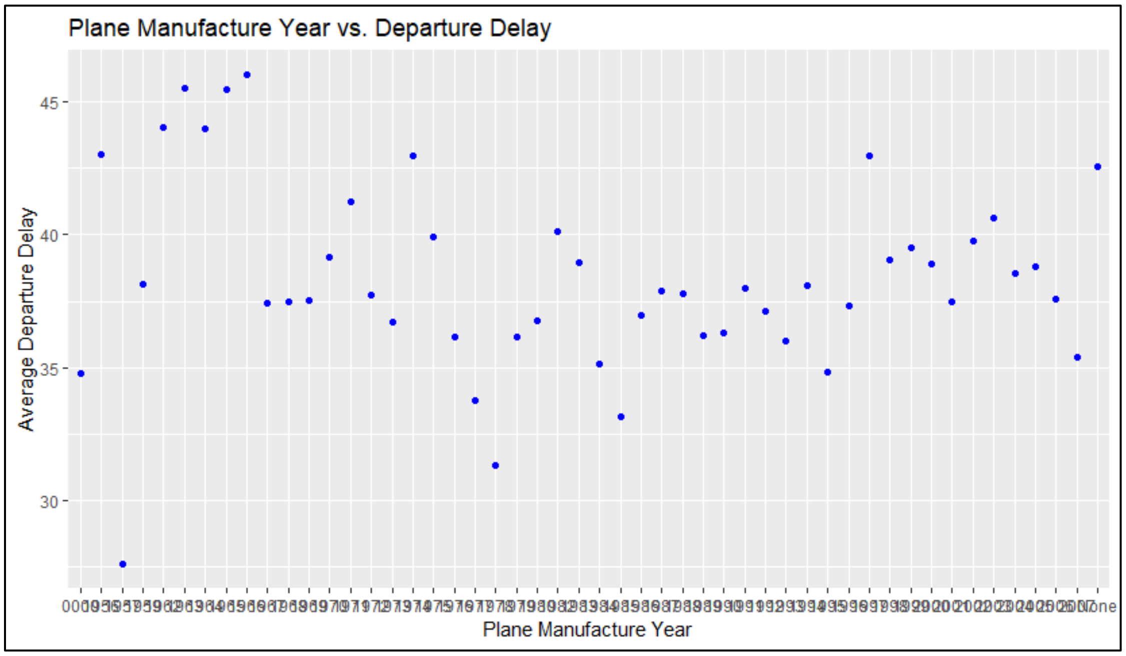


Figure 8: Departure Delay in R

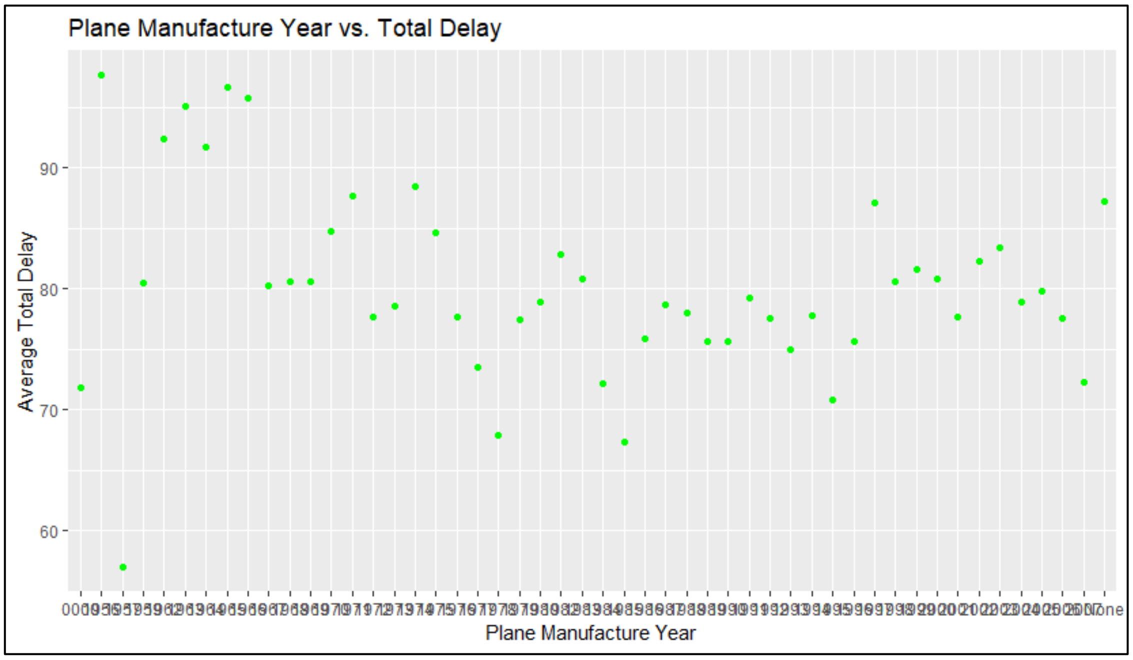


Figure 9: Total Delay in R

From the graphs above, it can be seen that there is a slight downward trend between the two variables, and the older planes do have a higher average departure/total delay. Therefore, it can be concluded that older planes do suffer more delays.

Question 3

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

To investigate how the number of people flying over different locations change over time, a count of the variable “FlightNum” was taken to find the number of flights each month for 2006 and 2007 separately. Then, two bar graphs were made to show this, for 2006 and 2007 respectively. A line plot was also created to show compare both years together.

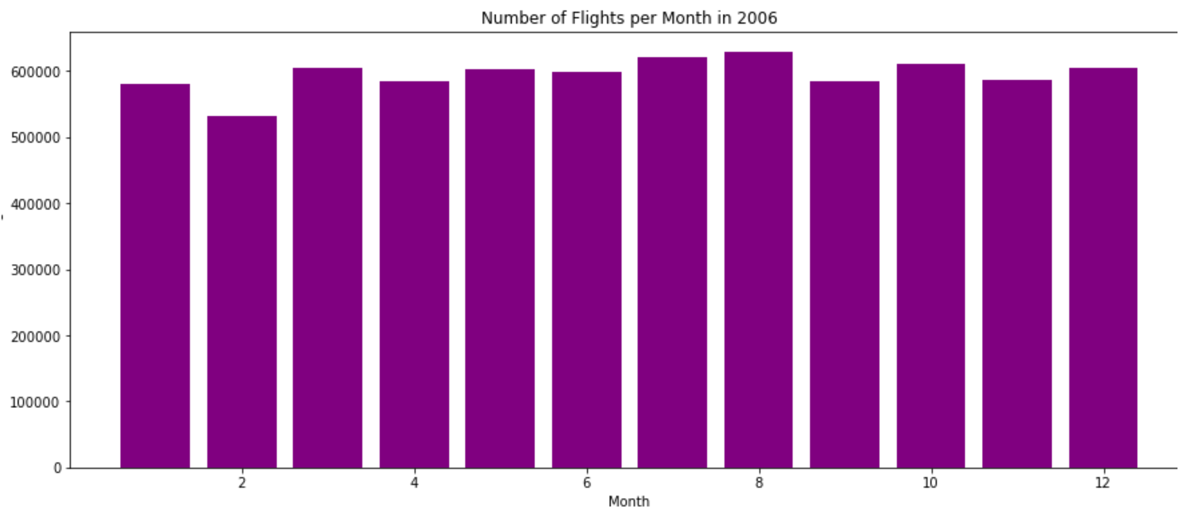


Figure 10: 2006 Flight Number - Python

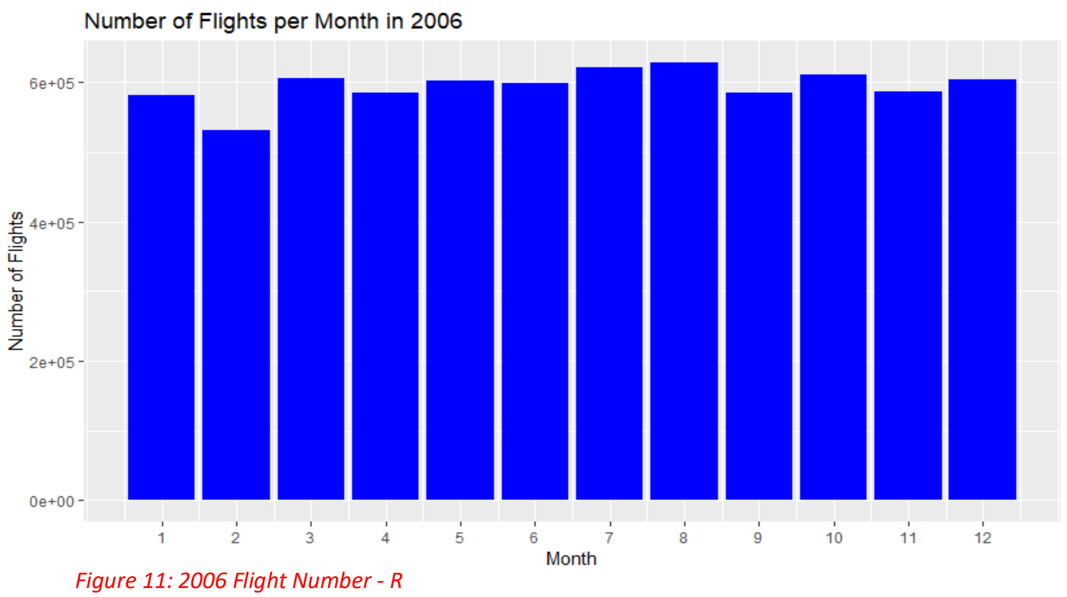


Figure 11: 2006 Flight Number - R

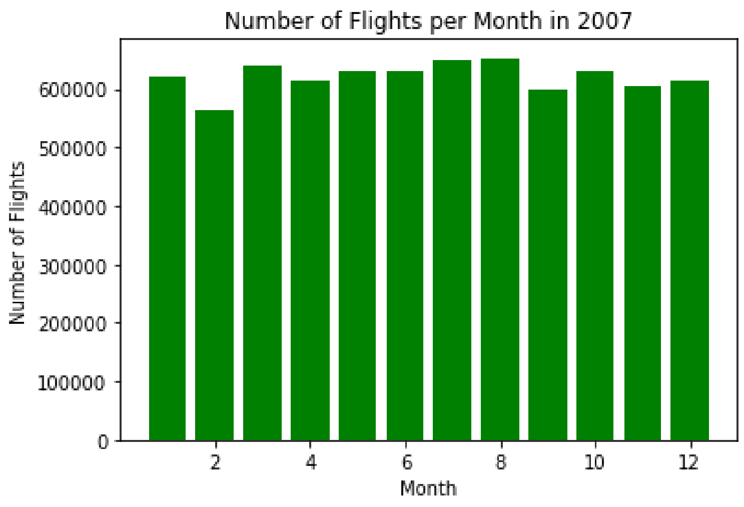


Figure 12: 2007 Flight Number per Month - Python

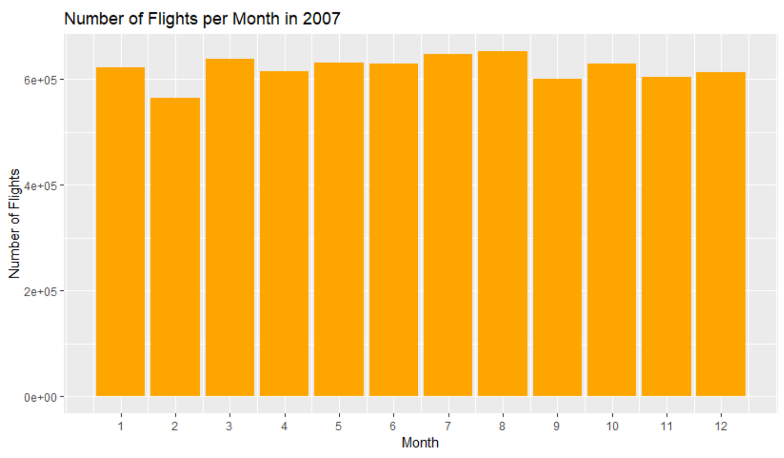


Figure 13: 2007 Flight Number per Month - R

From the graphs above, it can be seen that the number of flights decrease considerably in February and then increase and remain above 600,000 until September. Further, August seems to be the month with the highest number of flights in both 2006 and 2007.

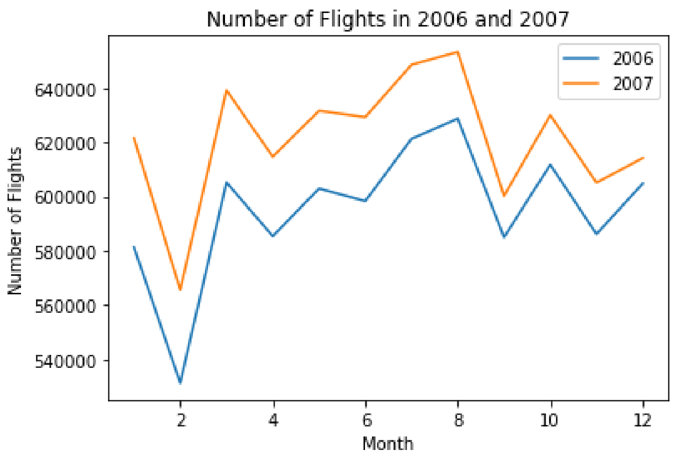


Figure 14: Line plot in Python

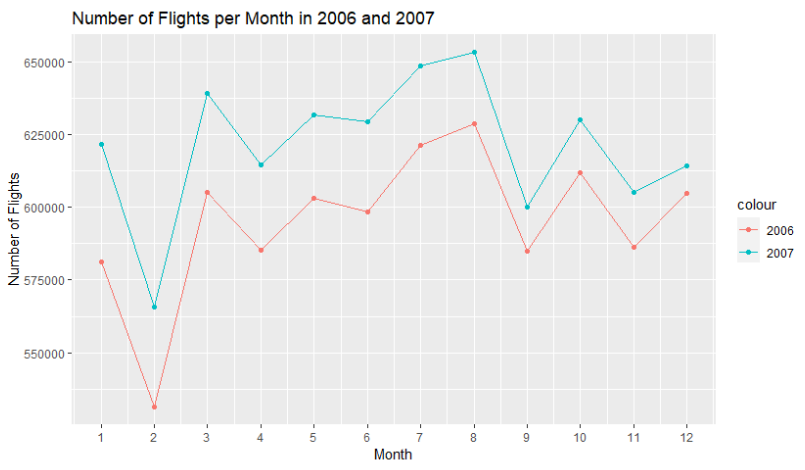


Figure 15: Line plot in R

From the line plots above, the same analysis can be made, that February has the lowest number of flights and August has the highest number.

A further analysis was done to show whether the number of flights changed by the season of the year. To do this, a variable was mapped out, matching each month to its respective season, and then a new column was created using this. A table was created to compare the season with the number of flights, and a bar graph was created using this.

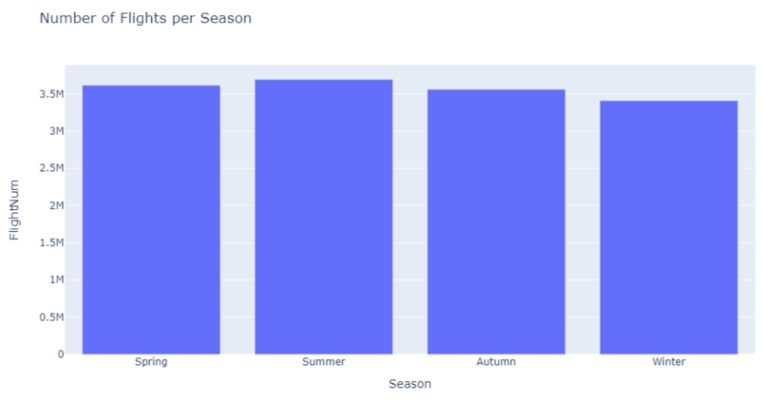


Figure 16: Seasonal Flight Number plot in Python

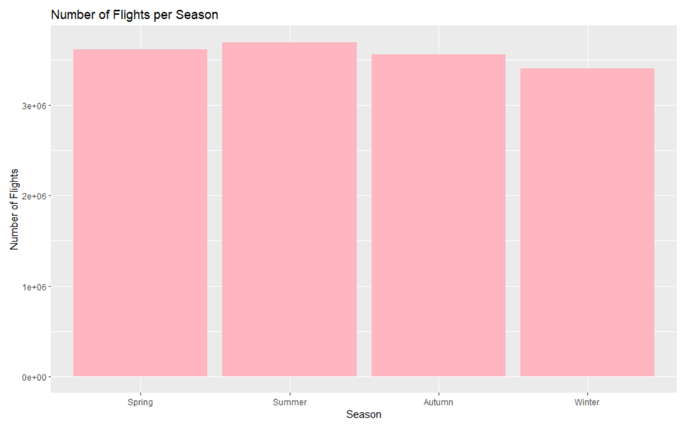


Figure 17: Seasonal Flight Number plot in R

From the graphs above, it can be seen that summer has the highest number of flights while winter has the lowest.

Question 4

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

To analyse this question, a variable was first created called "Day", taking the "DayofMonth" variable as a string. Next, the "CRSDepTime" variable was converted into date-time format using the hours and minutes extracted during the data cleaning process. Then, a variable called "TotalDelay" was created by adding the departure and arrival delays together. A similar column was then formed called "PreviousDelay" which returns the previous total delay grouped by "TailNum" (tail-number) and the "CRSDepTime" that was transformed to date-time format.

A scatter plot was then formed to compare current delay (total delay) with the previous delay, as shown below.

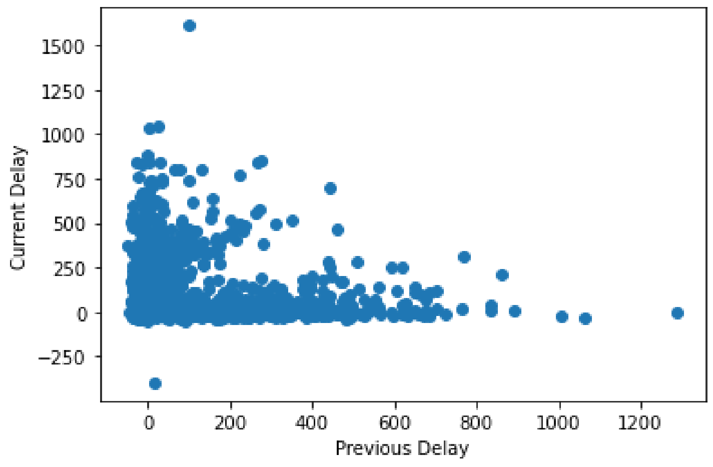


Figure 18: Previous Delay vs. Current Delay – Python

As shown by Figure 18, there is evidence that previous delays affect current delays, as when previous delay rises, the current delay seems to fall.

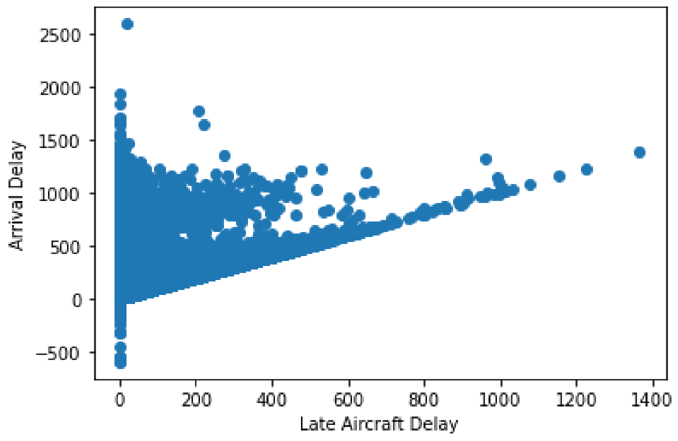
Currrent Delay	**0**	**1**
Previous Delay		
0	7930487	6663095
1	796	759

Currrent Delay	**0**	**1**
Previous Delay		
0	54.342292	45.657708
1	51.189711	48.810289

From the tables above, it can be seen that 759 flights face both a previous delay and a current delay, and about 49% of flights with a previous delay also face a current delay.

Another scatter plot was made to compare the "LateAircraftDelay" variable (arrival delay at an airport due to the late arrival of the same aircraft at a previous airport) with "ArrDelay" (arrival delay) to determine whether they had a strong correlation.

Figure 19: Late Aircraft Delay vs. Arrival Delay - Python



From the figure above, it can be seen that there is a strong positive relationship between Late Aircraft Delay and Arrival Delay, showing that delays in one airport do in fact cause further delays.

Another two variables were also created, to detect where a Late Aircraft Delay and Arrival Delay are taking place, and assigning 1 to those instances (and 0 if not delay takes place). A crosstab was then made using this, as shown below. Another table was also made to display the percentages of these events.

Arrival Delay	**0**	**1**
Late Aircraft Delay		
0	7944316	5080898
1	0	1569923

Arrival Delay	**0**	**1**
Late Aircraft Delay		
0	60.991827	39.008173
1	0.000000	100.000000

It is observed that when there is a late aircraft delay, there are no instances of having no arrival delay. Further, approximately 60% of flights which have no late aircraft delay will also have no arrival delay.

With each of these scatter plots and tables, it can be concluded that delays in one airport do cause delays in others.

Question 5

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

To construct a model, a sample of 300,000 records was first taken of the dataset that was merged in Question 2. A sample was taken because the codes would not run otherwise, as the size was too large to process. Next, the null values were removed from the columns containing such values.

Then, the process of setting up the pipeline was started. The numerical and categorical features of the dataset were chosen. For the numerical features, imputation was applied on the missing values, meaning they will be replaced with either the mean or median of the observations of each variable. The same was done for the categorical features, and dummy variables are created. The two sets of features are then merged, creating a new pipeline called "data_transformer".

Next, building a model was begun. A learner is attached to the created pipeline, so that the machine learning processes can be started. Then, a new categorical variable was created called "Delay", which has a value of 1 if a departure delay is present and 0 otherwise. The data is then split into the train and test sets in order to train the pipeline.

Finally, an ROC (receiver operating characteristic) curve was plotted for both the logistic regression and gradient boosting, as shown below.

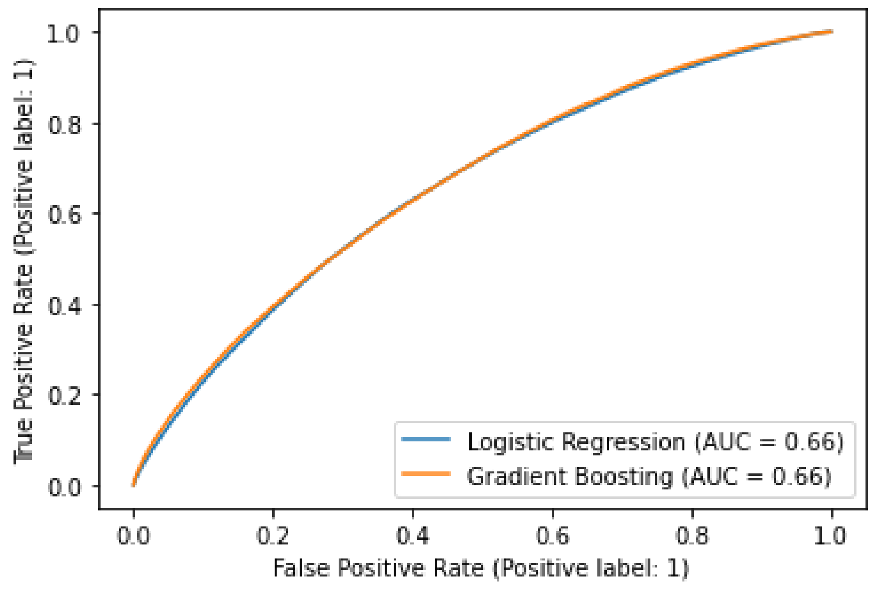


Figure 20: ROC curves in Python

As seen by the graph above, both the curves for logistic regression and gradient boosting give ROC values of 0.66, which is not a very sufficient value. The reason for this may be due to the small sample size of 300,000 values that was taken.