ABDUSSAMAD ROSHAN

PAGE COUNT – 10 (Excluding cover page, Table of contents, bibliography)

ST2195 COURSEWORK – ANALYSIS OF 2006 & 2007 FROM The 2009 ASA Statistical Computing and Graphics Data Expo

Table of Contents

[Introduction 2](#_Toc131432907)

[Data Cleaning 2](#_Toc131432908)

[When is the best time of day, day of the week, and time of year to fly to minimize delays? 3](#_Toc131432909)

[Best time of the year to fly to minimize delays. 3](#_Toc131432910)

[Best day of the week to fly to minimise delays. 3](#_Toc131432911)

[Best time of day to fly to minimize delays. 4](#_Toc131432912)

[Do older planes suffer more delays? 4](#_Toc131432913)

[How does the number of people flying between different locations change over time? 5](#_Toc131432914)

[Can you detect cascading failures as delays in one airport create delays in others? 6](#_Toc131432915)

[Use the available variables to construct a model that predicts delays. 8](#_Toc131432916)

[Bibliography 12](#_Toc131432917)

# Introduction

From October 1987 until April 2008, all commercial flights of all major USA based carriers were tracked for arrival and departure data. This data was made available from the 2009 ASA Statistical Computing and Graphics Data Expo. Two consecutive years, 2006 and 2007, were selected along with supporting data which contained information about the airports, carriers and planes and descriptions of the variables.

Before any question was attempted, a general data cleaning was done in both Python and R and the cleaned data was used to answer and solve the questions. Likewise, all questions have been done in Python and R ensuring replicability, where possible, for any testing.

This report contains the approaches taken to solve the questions, the visualizations obtained and the interpretation of the results.

# Data Cleaning

After importing and loading any necessary packages, the datasets for the years 2006 and 2007 were imported and checked for any differences that would interfere with the merging. Upon the observation of comparable number of columns with the same variables in each, the columns were merged with a new index. The merged data frame was checked for any errors during the merge by confirming the number of columns and rows in the new data frame.

The new data frame containing the data for both years was then checked for any duplicated rows, and having found the presence of multiple duplicated rows, all duplicates were removed keeping only the first instance.

Next, the data frame was checked for any null or empty values. There were multiple columns with null values present, the "CancellationCode" column stood out for having a very high number of null values. After investigating, it was found that it contained an incorrect entry. Both issues were resolved by removing the column completely as the column was not going to be of use in any future tasks. Then, all rows containing null values were removed from the data frame.

Next, the bounds of the data were checked, and having observed multiple entries of time greater than 2400, which is out of bounds for the standard 24-hour clock, the data frame was modified to only include rows where the “ArrtTime” (Arrival Time) and “DepTime” (Departure Time) had values less than 2400.

After all the above steps were implemented, the number of rows reduced from 14595137 to 14268895; a reduction of 2.24%. This was deemed acceptable, considering the size of the dataset.

Finally, the cleaned data was exported to a CSV file for use in future tasks.

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

For each part below, the necessary packages were imported and loaded for both Python and R.

## Best time of the year to fly to minimize delays.


Description automatically generatedChart, line chart

Description automatically generatedIn both Python and R, to calculate the average arrival delay for each month the “ArrDelay” was grouped by the “Month” column and the average calculated. A line graph was plotted to show how the average delay varied for each month.

Figure 1a - Average Arrival Delay for Each Month (Python)

Figure b - Average Arrival Delay for Each Month (R)

As can be seen, September and November have the lowest average delay of 6 mins. Their “ArrDelay” medians were thus compared to identify the better of the two.

The months of September and November carried median values of -3.0 mins and -2.0 mins respectively. A lower median in Arrival Delay is indicative of a better time and thus September is concluded to be the best time of the year.

## Best day of the week to fly to minimise delays.

Chart, line chart

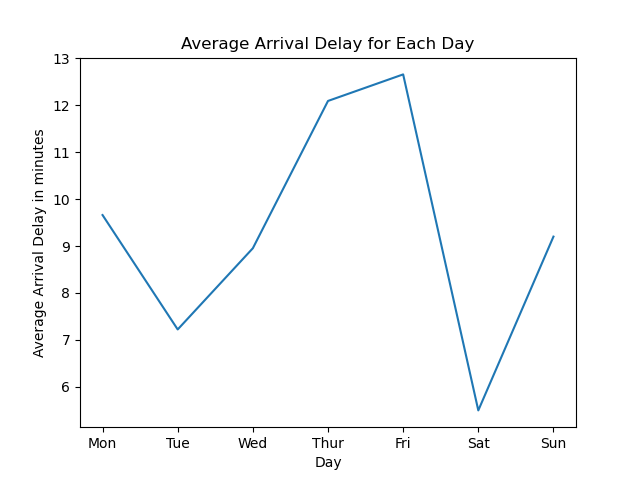
Description automatically generatedSimilar to finding the best time of year, in both Python and R, to calculate the average arrival delay for each day, the “ArrDelay” was grouped by the “Day” column and the average calculated. A line graph was plotted to show how the average delay varied against the day of the week.

Figure 2b - Average Arrival Delay for Day (R)

)

Figure a - Average Arrival Delay for Day (Python)

With regards to the above plots, Saturday has the lowest average delay of <6 mins and Friday has the highest; >12 mins. Thus, Saturday is the best day of the week to fly.

Furthermore, multiple studies done online are supportive of the result obtained, suggesting that this is the general trend in the USA.

## Best time of day to fly to minimise delays.

Time was split into 6 bins, each capturing a 4-hour time span in the 24-hour format. Average arrival delay was calculated based on the bins and plotted.

Chart, line chart

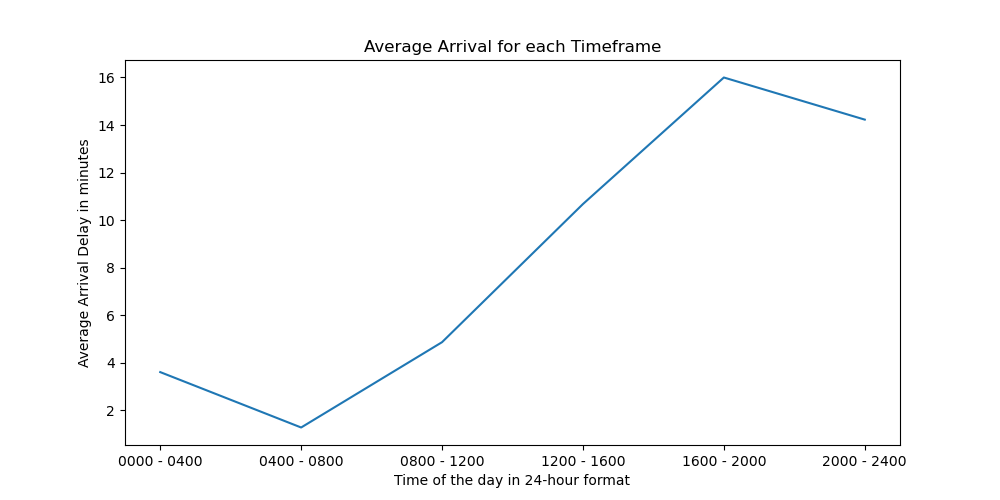
Description automatically generated

Figure 3a - Average Arrival Delay for Time (Python)

Figure b - Average Arrival Delay for Time (R)

As can be seen, the second bin; 4:00 am – 8:00 am has the lowest average arrival delay in mins, and hence the best time of the day to fly will fall within that range of time.

Delay increases exponentially from then on up to the fifth bin; 4:00 pm – 8:00 pm.

# Do older planes suffer more delays?

The plane-data file containing information about the planes was loaded and filtered to include only necessary information; “year” – the manufacture year of the plane and “talinum” – the tail number of the plane.

However, when checking the data types of the airports file it was noticed that the dtype of the "year" column was object which meant that there was erroneous data. Further investigation showed that there were values "0000" and "None" which were removed, and the data type of the "year" column was changed to integer.

The “talinum” column was renamed to “TailNum” to match the column name on the main dataset to facilitate the merging of datasets. The datasets were merged on the TailNum variable, and the merged dataset was checked for the existence of null values and removed.

A new variable, “mean\_ArrDelay\_byyear” was created to hold the mean average delay grouped by year. The year of manufacture of a plane and average arrival delay was compared with one another in the merged dataset, and thus plotted.

A scatter plot with a line of best fit was plotted for the year of manufacture of the plane against the average delay for planes manufactured in that year.

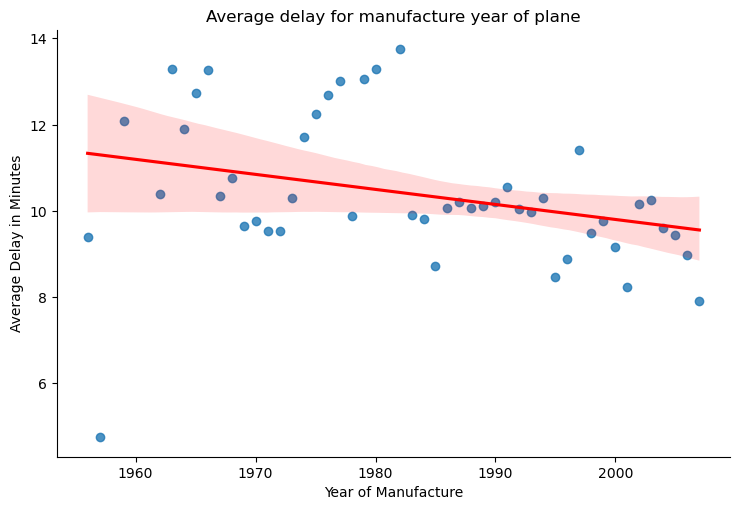
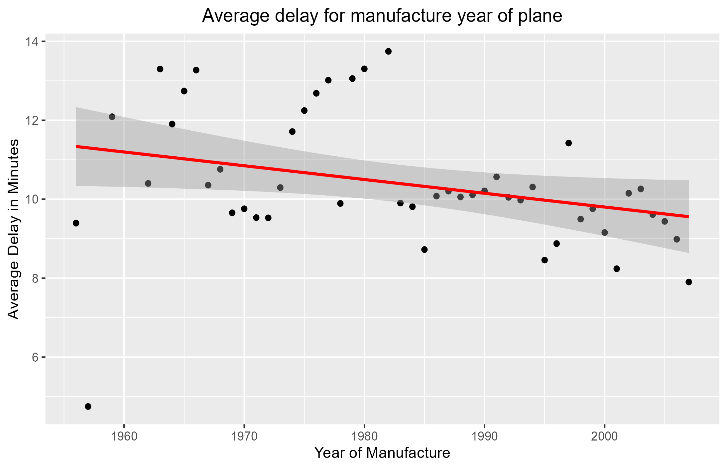


Figure b - Average Arrival Delay against the Year of Manufacture of planes (R)

Figure 4a - Average Arrival Delay against the Year of Manufacture of planes (Python)

A negative regression line can be observed, inferring that older planes suffer more delays compared to planes manufactured more recently.

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

The data-file consisting of information about the airports and their location, Airports.csv, was imported. The data frame was checked for duplicate rows and having found none, the existence of null values was examined and any rows containing them were removed from the dataset.

It was decided to use states to compare how the number of people flying between different locations changes over time. However, in the main dataset the origin and destination were given as their “iata”.

The International Air Transport Association's (IATA) Location Identifier is a unique 3-letter code (also commonly known as IATA code) used in aviation and logistics to identify an airport (Nationsonline.org).

Therefore, to get the origin and destination state certain steps were taken.

1. Columns were filtered in the main dataset to obtain a data frame with the necessary columns only; Year, Origin, Dest.
2. Similarly, the airports dataset was filtered to get a data frame containing only “iata” and “state”. This data frame was duplicated to obtain both origin and destination state for each flight. For one data frame, “iata” was renamed from “Origin” and “state” to “Origin\_State” and for the other data frame, “iata” was renamed from “Dest” and “state” to “Destination\_State”
3. The main dataset with the three columns, was first merged with one of the duplicated data frames using the “Origin” column to obtain a column containing the origin state for each flight, and then merged with the other data frame using the “Dest” column to obtain a column containing the destination state for each flight.

Having obtained columns containing the origin and destination state for each flight, the two columns, “Origin\_State” and “Destination\_State”, were combined to create a new column; “Route”, that captures the route of the flight from its origin to its departure state.

Using the “Route” column, the counts of flights for each route were obtained and the top 20 most flown routes were selected. Finally, a data frame containing the years, route and counts for the 20 most flown routes was obtained.

A bar plot indicating the total number of flights for the top 20 most flown routes for the years 2006 and 2007 was constructed.

Chart, bar chart

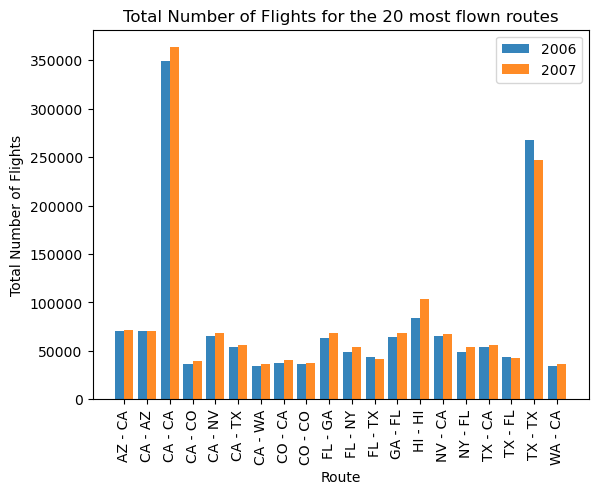
Description automatically generated

Figure b - Number of Flights for both years for 20 most flown routes (R)

Figure 5a – Number of Flights for both years for 20 most flown routes (Python)

As per the graphs, only a few routes have a significant difference over the years. “CA-CA” and “TX-TX” have a higher number of flights compared to the others in the Top 20.

“CA-CA” and “HI-HI” have higher number of flights in 2007 than 2006, whilst “TX-TX” had a lower number of flights in 2007.

In generality, the number of flights per route are approximately unchanged in the years 2006 and 2007.

Although the above analysis was only done for the top 20 routes, it can be replicated for all routes for a further in-depth analysis.

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

After importing the datasets, it was observed that the time of flights needed to be arranged in order. As such, “CRSDepTime” was used, with the values converted to a date-time format, so that they can be grouped by time.

In Python, extracting the first two numbers produced the ‘minutes’ part of the time, and the latter two values produced the ‘hour’ part of the time.

However, empty values were present for the hour portion since time below 60 minutes did not carry a value for hour. For instance, 30 which indicates 00:30 am does not carry with it an hour value. Therefore, all such empty entries were replaced with 0 to avoid errors.

The day of the month was obtained from the “DayofMonth” column. A lambda function was applied to iterate through each row and add a 0 to all single digit entries to ensure a consistent format in the values across the column. For instance, a value of ‘1’ would be formatted to ‘01’ upon the application of the lambda function.

By making use of the “CRSDepTime”, “DayofMonth”, “Month”, and “Year” columns, a new column; “DateTimeCRSDepTime” was created to obtain the scheduled departure time of each flight in data time format.

In R, the conversion to date time was handled differently. POSIXct conversion was used to store date and time in units of seconds. This benefits larger datasets as it reduces the processing power needed and the time taken to compute.

The rest of the steps were handled similarly as implemented in Python.

“TotalDelay”, a combination of arrival and departure delay, was computed to capture the entirety of delays because:

1. When a plane arrives late, the airport must deal with "ArrDelay"

2. Although a plane departs late, if it arrives on time, the departure delay is negated

3. Departure delay causes issues with air traffic control and scheduling conflicts

The flights were sorted by "DateTimeCRSDepTime" to get them ordered in a continuous timeline.

A new column, "PreviousDelay" was created by shifting the "TotalDelay" column down by one row, after grouping them by tail number. That is, each row will have the "TotalDelay" value of the previous row, allowing for the comparison between the delay of a particular flight taken from "TotalDelay", with the delay of the previous flight for the same aircraft, taken from "PreviousDelay", and thus the effect that the previous delay of a plane has on the Total Delay of its next flight can be observed.

The data frame was checked for null values, which were found to be present int the “PreviousDelay” column. There are multiple reasons for the existence of null values:

1. If a plane has had only one recorded flight, it will not have a "PreviousDelay"

2. The first flight for any individual plane will not have any "PreviousDelay"

The addition of columns and computed variables was done so that the effect of a preceding delay in one flight could be compared with the delay of the subsequent flight. This in essence is the cascading effect of delays caused from one airline to another.

Chart, scatter chart

Description automatically generatedA scatter plot with a regression line was plotted to show how "TotalDelay" and "PreviousDelay" are related.

Figure 6b – Scatter Plot for Previous vs. Current Delay (R)

Chart, scatter chart

Description automatically generated  
There is a slight upward line, indicating a positive relationship amongst the two variables. However, the slope is not steep suggesting that the relationship is not one that is strong, which is further supported with a correlation coefficient of **0.42.**

Figure 6a – Scatter Plot for Previous vs. Current Delay (Python)

This was further examined using cross tabulation.

Two new columns were created, with the inputs of absent or present depending on:

1. PresenceOfCurrentDelay - Present if "Totaldelay" greater than 0, Absent otherwise.
2. PresenceOfPreviousDelay - Present if "PreviousDelay" greater than 0, Absent otherwise.

Text

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generatedA cross tabulation was made, with the values converted to percentages to gather how previous delay affects the current delay as a percentage value.

Figure 7b – Cross Tabulation (R)

Figure 7a – Cross Tabulation (Python)

The above figures indicate that only 31.39% of current delays existed when there were no delays in the previous flight. However, there were 64.09% current delays when there had been prior delays. This shows that the existence of previous delays causes relatively more current delays. Consequently, this demonstrates a cascading effect in delays between airports.

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

Flight delays triggers a negative turn of events for the airport industry as it stimulates operation inefficiencies, fuels up capital costs due to the need to reallocate aircrafts and crew members, and builds up negative externalities as fuel consumption increases exponentially.  
These undesirable impacts of delays can be prevented by forecasting the delays to restructure operations in advance. In specific, the arrival delay will be predicted since this would allow for the destination airport to account for any lags in flights that may happen and thus dynamically prepare to face them without much concern.

Essential packages, main dataset and the airports dataset were imported in Python and R.

As mentioned above, ArrDelay will be our target variable to predict the delays. As such, a new column “PrecenseOfArrivalDelay” was created in the main dataset to capture a binary view of the column. This was computed by running a lambda function to display “Present” if “ArrDelay” was greater than 0, and “Absent” otherwise.

The Plane dataset was filtered to include only the necessary columns of “tailnum” and “Year”. The dataset was thereafter checked for duplicate rows and having found none, the existence of null values was assessed and any rows containing them were removed from the dataset.’

Further, it was noted that “Year” was of data type ‘object’ instead of ‘integer’ and hence, required a conversion by dropping off ‘None’ and ‘000’ from the “Year” column.

Prior to the merging of the main dataset and the Plane dataset, the “talinum” column had to be renamed to “TailNum”, so that there existed no change in column names between the two data frames. After renaming the column, the data frames were merged on the “TailNum” column.  
For a more efficient interpretation, “Year” in the merged data frame was renamed to “YearOfManufacture”.

A picture containing chart

Description automatically generatedA Pearson correlation matrix was constructed to visualize the correlation between variables.

Chart

Description automatically generated with medium confidence

Figure 8b - Pearson Correlation Matrix for Variables (R)

Figure 8a – Pearson Correlation Matrix for Variables (Python)

Upon obtaining the value counts for “Cancelled” and “Diverted”, it was observed that they only contained zeros, and hence was excluded from the model.

The features that will be used to predict “PresenceOfArrivalDelay” are "Month", "DayOfWeek", "DepDelay", "Origin", "Dest", "Distance", "TaxiOut", of which

* “Distance", "DepDelay", "TaxiOut" are numerical features, and
* “Month", "DayOfWeek", "Origin", "Dest" are categorical features

Most of the predictor variables were chosen as the showed a significant relationship in some way with Delays in the previous questions, and hence would be helpful in explaining and predicting a delay. While it is understood that “DepDelay”, departure delay is highly correlated with the delay itself, the variable was selected under the assumption that airline systems are integrated with a free flow of information with regards to scheduled flight timings, flight departure and arrivals (by extension, delays), and hence “DepDelay” will be available at the time of take-off from the Origin airport as well.

“CRSDepTime” was considered and used as a numerical variable, although it is a time-based value by nature, to ensure effectiveness in the model as converting it to a categorical form would birth numerous categories, increasing computation time.

Before the next steps could be taken, the data frame had to be reduced in sizein R as it faced a memory failure when running the complete dataset. As such, a random sample of 10% was taken after setting seed to ensure replicability.

A data frame, “X”, was created in Python with the required features, mentioned above. Following “X”, a Pipeline was built to:

* Scale numerical data and impute any missing values, numerical transformer pipeline.
* Encode categorical variables and impute any missing values, categorical transformer pipeline.

The transformer pipelines were combined to get a data transformer, which would enable the application of a function to a set of points.

A pipeline was then created with the data transformer and logistic regression as the estimator, to allow the computations of multiple operations in a sequential order, so that the output of each operation will flow through as the input for the next operation (Kozak, 2022).

The dataset was split, with a 70:30 proportion, into training (70) and test (30) data sets and a random seed was set to ensure replicability.

Parameter grid, which comprises of a discrete number of values for each parameter variable, was set and hyperparameters were tuned using GridSearch. The logistic regression model was fitted using the specified hyperparameters and training configuration.

Chart, scatter chart

Description automatically generatedChart

Description automatically generated with medium confidenceAn ROC curve (receiver operating characteristic curve) is a visual display of the performance of the classification; logistic regression, model at all classification thresholds, plotting the ‘True Positive Rate’ and ‘False Positive Rate’ parameters. The ROC curve was plotted to evaluate the model.

Figure 9b – ROC Curve (R)

Figure 9a – ROC Curve (Python)

Chart

Description automatically generatedChart, treemap chart

Description automatically generatedArea under curve (AUC) is an accuracy metric for ROC. (Vidhya, 2015). AUC for the model in both Python and R was 0.91, indicating that the model had a very good performance rate (Stephen Allwright, 2022).

Figure 10b – Confusion Matrix (R)

Figure 10b – Confusion Matrix (Python)

Similarly, a confusion matrix was computed as a performance measurement for the model. Confusion Matrix consists of combinations of predicted and actual values, each of which measures related but different values (Narkhede, 2021).

An accuracy score of 0.83 was obtained in Python based on the confusion matrix using ‘sklearn’. In R, a similar score was computed by:

* (123669+163343)/(123669+163343+17999+39732) = 0.83253902182 ≈ 0.83

An accuracy of 0.83 in both Python and R suggests that the model is significantly accurate.

# Bibliography

*Classification: Roc curve and AUC  |  machine learning  |  google developers* (no date) *Google*. Google. Available at: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc#:~:text=An%20ROC%20curve%20(receiver%20operating,False%20Positive%20Rate (Accessed: April 1, 2023).

*The impact of flight delays on passenger demand and societal welfare* (no date). Available at: https://www.researchgate.net/publication/251670916\_The\_impact\_of\_flight\_delays\_on\_passenger\_demand\_and\_societal\_welfare (Accessed: April 1, 2023).

Kozak, M. (2022) *Building Generator Pipelines in python*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/building-generator-pipelines-in-python-8931535792ff (Accessed: April 1, 2023).

Narkhede, S. (2021) *Understanding confusion matrix*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 (Accessed: April 1, 2023).

Stephen Allwright (2022) *How to interpret AUC score (simply explained)*, *Stephen Allwright*. Stephen Allwright. Available at: https://stephenallwright.com/interpret-auc-score/ (Accessed: April 1, 2023).

Vidhya, A., 2015. Logistic Regression R | Introduction to Logistic Regression. [online] Analytics Vidhya. (Accessed: April 1, 2023)