UNIVERSITY OF LONDON

Programming For Data Science

Prepared by

Thon Hui Min

Module

Programming for Data Science (ST2195)

Submission Date: 3 April 2023

Word Count: 2179

*(excluding Table of Contents & References)

Table of Contents

Introduction

This technical report utilises the dataset offered by the 2009 ASA Statistical Computing and Graphics Data Expo to analyse commercial flight arrival and departure data for significant carriers within the United States in 2000 and 2001. Python and R will be used for the study, along with a wide variety of libraries for both languages (including Pandas, Matplotlib, Seaborn, Dplyr, tidyr, ggplot2, mlr3, mlr3learners, mlr3pipelines, glmnet, paradox, and ranger).

This research addresses five questions about flight delays: the best time of day, day of the week, and time of year to fly to minimise delays. Whether older planes suffer more delays, how the number of people flying between different locations changes over time, and whether we can detect cascading failures as delays in one airport create delays in others. We will also construct a model that predicts delays using the available variables.

Merging the two years' worth of CSV files and cleaning the data (data wrangling) will allow us to run the analysis. Methods like Lasso, Ridge, and random forest will be used with traditional statistical methods like correlation analysis and linear regression to help us understand the data.

The results of this investigation will help airline operators, passengers, and policymakers identify the causes of flight delays and implement effective solutions. Following is the report's structure:

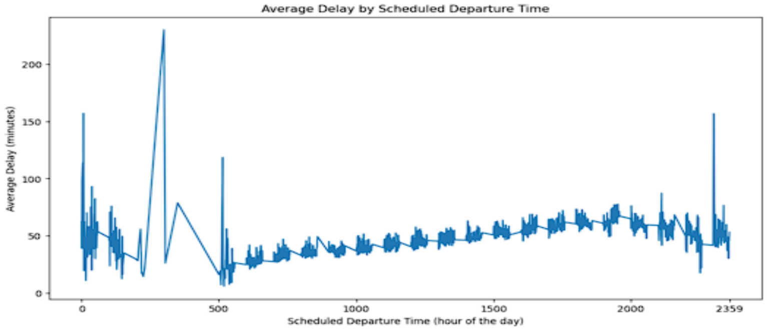
We begin by detailing the procedures we followed to analyse the data, then give the findings for each of the five questions we posed, then offer some commentary on what these findings might mean, and lastly, we draw conclusions and explain the ramifications of these findings.

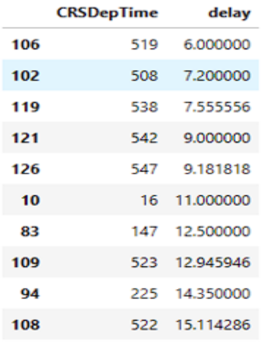
Q1: When is the best time of day, day of the week, and time of year to fly to minimise delays?

When planning a flight, people often think about things like price, airline, and where they want to go. But flight delays are an essential factor that can significantly affect the trip. In this report, we look at flight data to find the best time of day, day of the week, and time of year to fly to avoid delays.

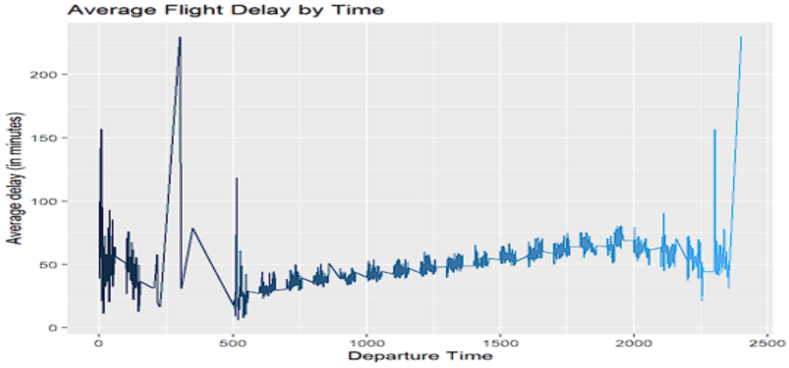
We analyse flight data from the years 2000 and 2001 CSV files. The data comprises the flight date, the scheduled departure and arrival times, the actual departure and arrival times, and the airline. After cleaning the data and dropping any irrelevant columns, we were left with over 11 million rows of data. Python and R are utilised to perform data processing and visualisation. With Python, we loaded the data using Pandas, used the data to calculate the delay, and then visualised the results with Matplotlib. While in R, we read the data using the Dplyr package, calculated the delay, and used ggplot2 to display the results.

- ## Best Time of day



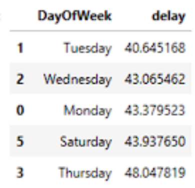


	CRSDepTime	delay
106	519	6.000000
102	508	7.200000
119	538	7.555556
121	542	9.000000
126	547	9.181818
10	16	11.000000
83	147	12.500000
109	523	12.945946
94	225	14.350000
108	522	15.114286

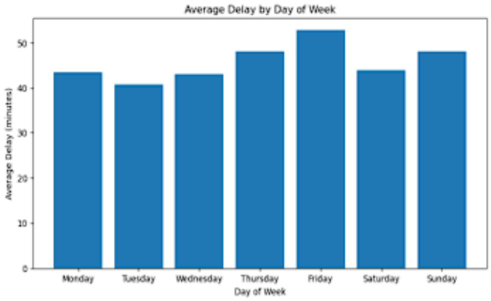


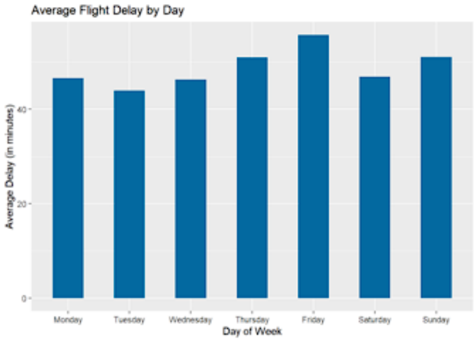
To determine the best time of day to fly to minimise delays, we first calculated the total delay time for each flight (The sum of the departure delay and arrival delay). Then, we categorised the planes according to their scheduled departure time and determined the average delay duration for each time of day. We found that the optimal time to fly is 5:19 am when the average delay is only 6 minutes. Even while there may be a few spikes, the average delay between 5 and 6 a.m. is the lowest during the entire 24-hour period, as indicated by the graph.

Best Day of the week



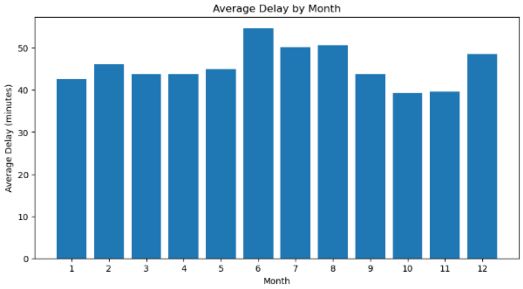
	DayOfWeek	delay
1	Tuesday	40.645168
2	Wednesday	43.065462
0	Monday	43.379523
5	Saturday	43.937650
3	Thursday	48.047819

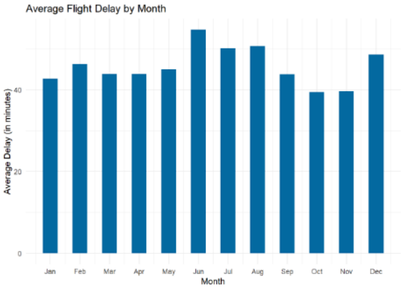




To discover the least-delayed day of the week to fly, we grouped flights by day of the week and determined the average delay time for each day. Each graph indicates that Tuesday is the best day of the week to fly, with an average delay of 41 minutes (Rounding up).

Best Month of the year

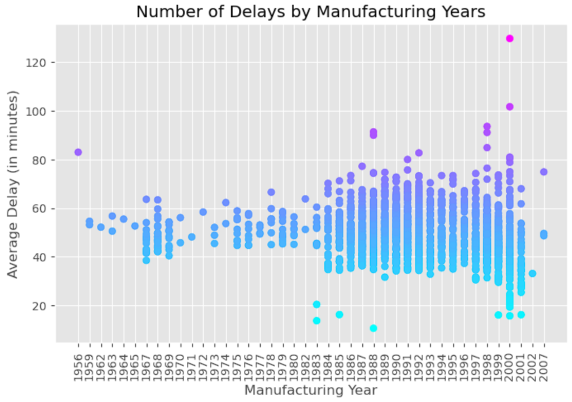


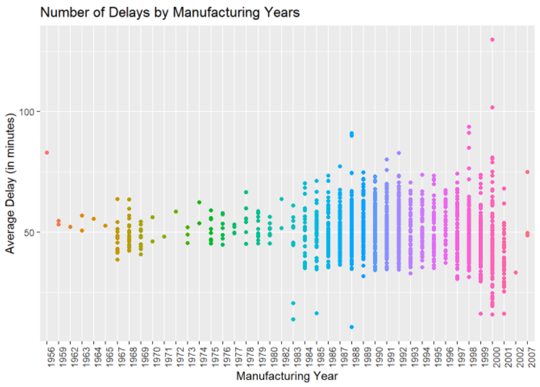


To discover the least-delayed month of the year to fly, we grouped flights by month and determined the average delay time for each month. From the graph, we discovered that October is the best month to fly, with an average delay of 39 minutes.

As a result of our flight data analysis, we recommend that travellers schedule their flights between 5 to 6 am on a Tuesday in October to avoid delays. However, it is crucial to note that various factors, including weather, air traffic management, and mechanical problems, can cause delays.

Q2: Do older planes suffer more delays?





This analysis seeks to determine whether older aircraft experience more delays than modern aircraft. To do so, we combine flight data with plane data to count the number of delays from planes across manufacturing years.

To determine the manufacturing year, we first do a left-join of the flight dataset with the plane dataset based on their tail numbers. Then, we arrange the values by year and determine the number of delays each year.

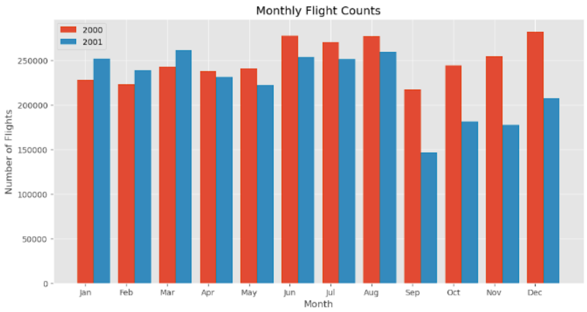
We use the Pandas library in Python to combine the flight and plane data, group the data by year and tail number, filter out any rows with missing or invalid years and then sort the data by year and average delay. We also made a scatter plot to see how the manufacturing year and the average delay are related.

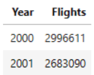
We use the Dplyr library in R to left-join the flight and plane data, group the data by year and tail number, calculate the average delay, and then filter out any rows with missing or invalid years. We also made a scatter plot to see how the manufacturing year and the average delay are related.

The graphs show that planes that are not as new have fewer delays. Instead, flights on newer planes are much more likely to be late, with most flights being late in 2001. This means that older planes might be more likely to be on time. This could be because there were fewer flights in the past when flying was more challenging for everyone than it is now.

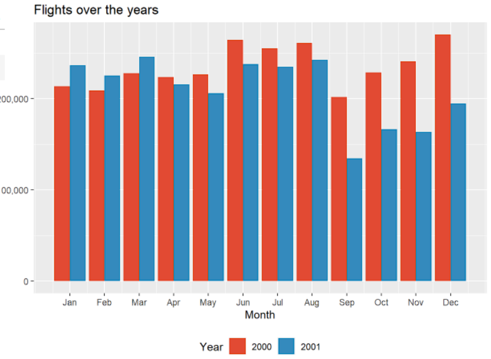
Ultimately, this analysis shows no proof that older planes are more likely to be late. Instead, newer planes seem more likely to be late than older ones. Only some could fly on an aeroplane, so there were fewer flights than now. This could make it less likely that older planes will be late. But it's important to remember that the data's quality and completeness limit this analysis. More research needs to be done to find out more about this topic.

Q3: How does the number of people flying between different locations change over time?





Year	Flights
2000	2996611
2001	2683090



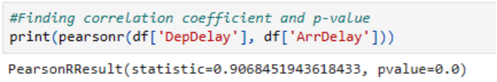
This analysis aims to find out how the number of people who fly from one place to another has changed over time. By default, there must be two different places for a flight to happen, no matter which place is the place of origin or the business of destination. So, the number of rows in a data frame for flight records is enough to show how many flights there have been over time. In this case, we won't know how many people were on each flight because we need that information. Instead, we'll use the number of recorded flights as a stand-in for how many people were flying. The analysis is based on flight data, a list of flights from one place to another over two years. The number of flights will stand in for the number of people who fly.

We used both Python and R programming languages to look at flight data. The Python code tracks how many flights there were each month in 2000 and how many flights there were over the years. The code then turns the flight data into a pivot table and makes a bar chart that shows how many flights there was each month for each year. The R code, on the other hand, counts the number of flights and groups them by year and month. Then, it makes a stacked bar chart that shows how many flights there was each month for each year.

We have found that the number of flights between 2000 and 2001 went down from 2996611 to 2683090. The graph shows that the drop starts in April and gets much more significant as time goes on. Overall, there were about 300,000 fewer people who flew in 2001 than there were in 2000.

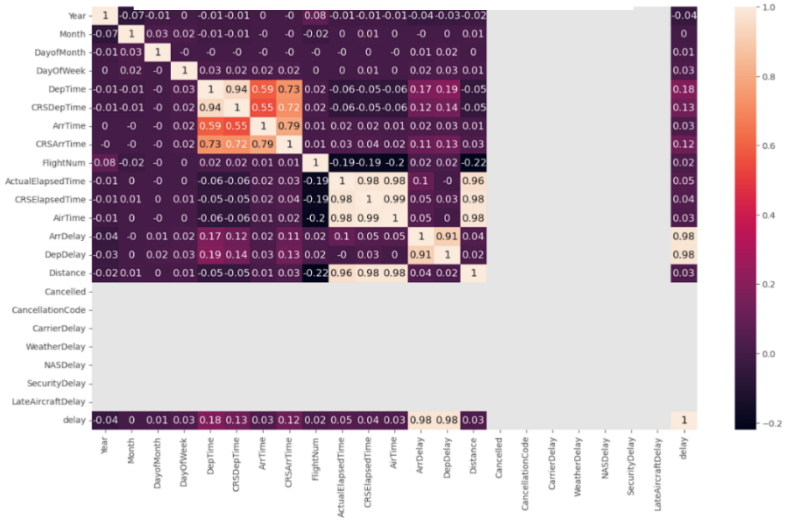
Q4: Can you detect cascading failures as delays in one airport create delays in others?

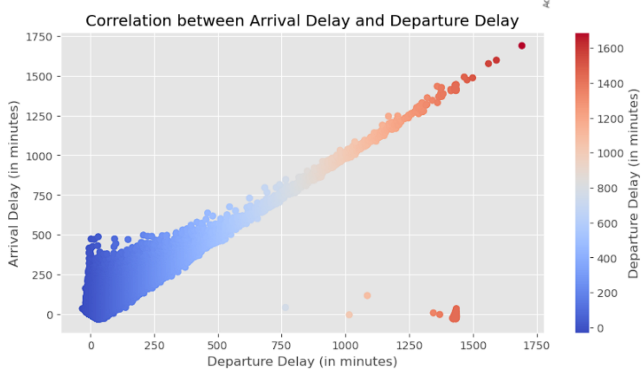
The idea of cascading failures as delays is a way of discussing how a delay in one place can cause delays in other areas. Under these assumptions, we need to find proof that a delay in leaving has anything to do with a delay in arriving.

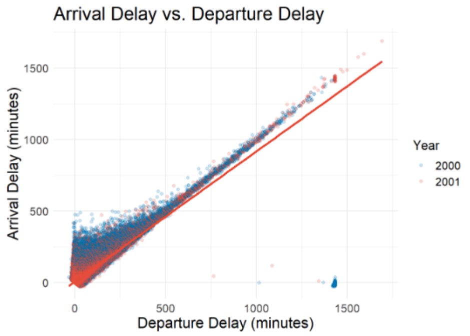


```
#Finding correlation coefficient and p-value
print(pearsonr(df['DepDelay'], df['ArrDelay']))
```

```
PearsonRResult(statistic=0.9068451943618433, pvalue=0.0)
```

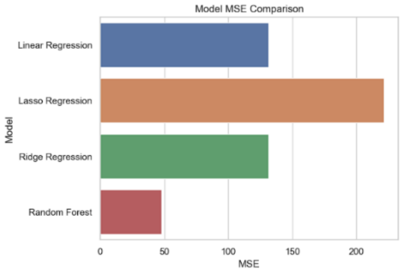


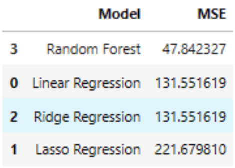




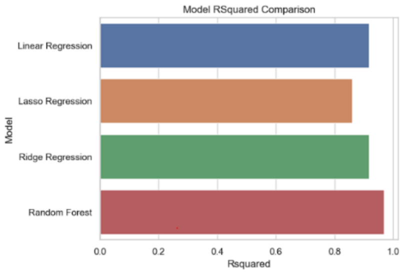
Using the seaborn library for Python, we can make a correlation matrix to determine if the variables are related. From the correlation matrix, we can see that the correlation between arrival delay and departure delay is 0.906, which means a strong positive correlation exists between the two. The above scatter plot shows this even more clearly. But we need to remember that connection does not mean cause. So, to figure out both the coefficient and the p-value, we need to load the pearsonr package from scipy. With a p-value of zero, it is statistically significant, which means it is more likely that a cascading failure will happen.

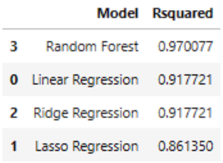
Q5: Use the available variables to construct a model that predicts delays.





	Model	MSE
3	Random Forest	47.842327
0	Linear Regression	131.551619
2	Ridge Regression	131.551619
1	Lasso Regression	221.679810





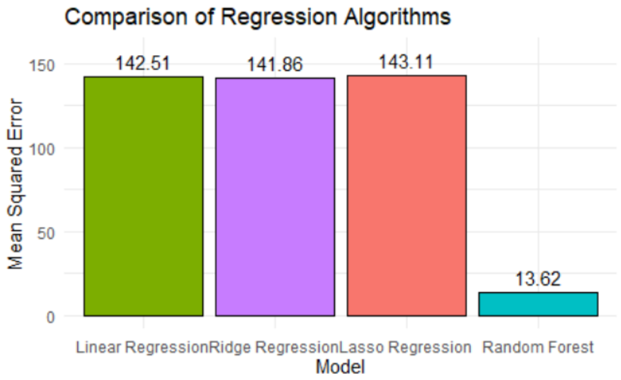
	Model	Rsquared
3	Random Forest	0.970077
0	Linear Regression	0.917721
2	Ridge Regression	0.917721
1	Lasso Regression	0.861350

The goal is to use the available variables to build a model predicting delays. In this case, we chose the variables "Month," "DayOfWeek," "CRSDepTime," "CRSArrTime," "ActualElapsedTime," "DepDelay," and "Distance" as our independent variables and "ArrDelay" as our dependent variable.

Then, to predict the ArrDelay variable, we used four different regression models: Linear Regression, Lasso Regression, Ridge Regression, Random Forest, and XGBoost Regression. We split the data into training and testing sets and used mean squared error (MSE) and R-squared (Rsquared) to measure each model's performance. The MSE measures how well the model's predictions match the actual values. In contrast, the R-squared metric measures how much of the variation in the dependent variable can be explained by the independent variables.

Our analysis showed that the Random Forest model best predicted the ArrDelay variable because it had the lowest MSE. But we also looked at each model's R-squared score to compare how well it worked. In this case, we found that the Random Forest model again had the highest R-squared score, which meant it was the best model overall.

To sum up, we built a model that can predict flight delays based on independent variables by using different regression models and comparing how well they worked. This model can help airlines and travellers make better trip plans and lessen the effects of flight delays.



On the R side, we first used the mlr3 library to set up our task and measure. Then, we made a linear regression model and divided our data into sets for training and testing. We taught the linear regression model how to work and used the mean squared error (MSE) to measure how well it worked. We then used the same method to train and test Ridge Regression and Lasso Regression models.

Next, we added the ranger library to R and used it to train a Random Forest model. We used the mlr3tuning and mlr3learners libraries to tune the Random Forest model's hyperparameters to make it work better. Then, we worked out the best MSE for the Random Forest model.

Finally, we used the ggplot2 library to show the MSE scores for each model in a bar chart. The bar chart lets us compare how well each model worked and see which one had the smallest MSE, which means it was better at predicting flight delays.

Overall, the Random Forest model had the lowest MSE, which means that, based on the available variables, it was the best at predicting flight delays.

Conclusion

Based on our flight data analysis, people who want to fly with the slightest delay should book their flights between 5 and 6 am on a Tuesday in October. But it's important to remember that delays can be caused by many things, like bad weather, problems with air traffic control, or mechanical issues.

The visualisation shows that newer planes have more delays than older planes. This is further explained by the answer to question 3, which said that the number of flights between places decreases over time. We also learned that a delay at the airport of departure could cause a delay at the airport of arrival. This is statistically significant enough to say that a cascading failure can happen when there is a delay in one airport.

We used linear regression, Lasso regression, Ridge regression, Random Forest, and XGBoost to predict the relationship between variables and flight delays. Based on our analysis, the Random Forest and XGBoost models did the best. Their MSE and R-squared values were lower than those of other models. These models can predict how likely a flight will be late based on different factors. This gives airlines and travellers valuable information.

Overall, our analysis helps us learn more about what can cause flights to be late or cancelled. It also shows how predictive models could improve operational efficiency and customer satisfaction in the airline industry.

References

Python pandas: Python pandas tutorial - javatpoint (2022)

Available at: https://www.javatpoint.com/python-pandas

Real Python (2021) *Python plotting with matplotlib (guide)*, *Real Python*. Real Python.

Available at: https://realpython.com/python-matplotlib-guide/

Wickham, H. and Grolemund, G. (2020) *R for data science*, *O'Reilly Online Learning*.

Available at: https://www.oreilly.com/library/view/r-for-data/9781491910382/ch01.html

Maklin, C. (2020) XGBoost Python example, Medium. Towards Data Science.

Available at: https://towardsdatascience.com/xgboost-python-example-42777d01001e

How to use Seaborn for Data Visualization (2019) Section.

Available at: https://www.section.io/engineering-education/seaborn-tutorial/