ST2195: Programming for Data Science coursework

Student no.: 200217868

Table of Contents

1) Introduction

Airline travel is an immensely popular mode of transportation, with billions of people taking it every year [1]. In 2019, there were approximately 38.3 million flights worldwide, and while the COVID-19 pandemic caused a significant dip in air travel, the numbers are expected to rebound [1]. Despite increased efficiency since the start of flight travel, flight delays remain a persistent issue for travellers. Through data exploration and modelling, this study aims to analyse trends in flight delays, connections, and other factors, providing insight into how flight data evolves over time.

2) Data

The data is from the Data Expo 2009: Airline on time data found in the Harvard Dataverse [2]. The data has 29 variables detailing each flight. When reading the data for both Python and R, the files for the code should be in the data folder or the working directory must be set to where the data is located.

3) Data pre-processing

To start with, for the purposes of answering the specific questions within this report, most of the columns are not needed. Hence, only 13 variables will be used in the main data set that will be used for analysis. These variables and their descriptions are as listed:

Name	Descriptions
1) Year	1987 – 2008
2) Month	1 – 12
3) DayOfWeek	1 – 31
4) DepTime	1 (Monday) – 7 (Sunday)
5) CRSDepTime	actual departure time (local, hhmm)
6) ArrTime	scheduled departure time (local, hhmm)
7) CRSArrTime	actual arrival time (local, hhmm)
8) TailNum	plane tail number
9) ArrDelay	in minutes
10) DepDelay	in minutes
11) Origin	in minutes
12) Dest	arrival delay, in minutes
13) Year of manufacture	the year the plane was manufactured

Table 1: Selected variables

The reason for this is because these variables were the only ones determined to be crucial in answering the questions posed. We can also see that it is quite a large dataset since it is reading 3.5 gigabytes even when only 12 variables were selected so determining the

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14269866 entries, 0 to 7140595
Data columns (total 12 columns):
 #   Column      Dtype
---  ------      -----
 0   Year        int64
 1   Month       int64
 2   DayOfWeek   int64
 3   DepTime     float64
 4   CRSDepTime  int64
 5   ArrTime     float64
 6   CRSArrTime  int64
 7   TailNum     object
 8   ArrDelay    float64
 9   DepDelay    float64
 10  Origin      object
 11  Dest        object
dtypes: float64(4), int64(5), object(3)
memory usage: 3.5 GB
```

Figure 1: Data info

```
Year          0.000000
Month         0.000000
DayOfWeek     0.000000
DepTime       1.832442
CRSDepTime    0.000000
ArrTime       2.027342
CRSArrTime    0.000000
TailNum       0.000890
ArrDelay      2.027342
DepDelay      1.832442
Origin        0.000000
Dest          0.000000
dtype: float64
```

Figure 2: NA value percentages

important variables and selecting only those helps with reducing memory errors. The last variable, year of manufacture, is from the plane-data file. The main dataset is first checked for NA/missing values. There are missing values in the variables 'DepTime', 'ArrTime', 'ArrDelay' and 'DepDelay. We can check how much NA values are there and see the percentage of them compared to the total amount of data. As a rule of thumb, dropping NA values that are less than 5% of the respective observations is acceptable [3]. As we can see, the variables that have NA values have them for less than 5% of their observations and hence, the NA values will be dropped.

Now that the data is set, we can start to pre-process the data.

For question 1, we start with the best time of the day. For determining what is the best time of the day to fly, the variables 'CRSDepTime' and 'CRSArrTime' were selected. This is due to the fact that we are looking for departure and arrival delays start at the scheduled departure and arrival timings. This would show the delays that are expected at a said scheduled time. For example, if the scheduled departure time is 06 hours (6 AM) and the mean departure delay is found out to be one hour then the observation is that there is an expected mean departure delay of an hour *after* 6 AM. If the actual departure time was taken, this would be opposite since the actual departure time would already consider the delay. If the actual departure time is 07 hours (7 AM) and the mean departure delay is the same, then the delay *ended* at 7 AM. Hence, the variables 'CRSDepTime' and 'CRSArrTime' were selected. Next, we have the day of the week. First, the weekly data frame is created by extracting the variables 'DayOfWeek', 'ArrDelay' and 'DepDelay'. This allows us to focus on just those variables for this part of the question and makes it easier to handle the data as well. The same goes for the monthly data frame where only 'Month', 'ArrDelay', and 'DepDelay'. For both weekly and monthly analysis, total delay, the sum of arrival and departure delay, was calculated due to the fact that on a timeframe like a week or as large as a month, looking at arrival and departure delay separately seems excessive. On such a timescale, there will be very nuanced changes between

the two types of delays especially on a monthly timescale and hence total delay was taken for both weekly and monthly analysis.

For question 2, the plane-data dataset was used. The NA values were checked and removed. The variable ‘TailNum’ was used which then allowed for the year of manufacture to be extracted.

For question 3, the data was split in two: pre-1980 year of manufacture and post-1980 year of manufacture. This is due to the sampling variations between the two periods. There are about 8 million observations in post-1980 compared to the 500,000 in pre-1980. Hence, it wouldn’t give an accurate representation if the two were not separated.

For question 4, two different data subsets were taken for Python and R. For Python, the top 5 connections, in ascending order, with total flights between 100 and 500 were analysed while for R, the top 5 connections, in ascending order, with total flights between 1000 and 2500 were analysed. This was done due to the fact that at higher levels of total flights, there is very little change since the total flights number is so large. Hence, to find a greater change between 2004 and 2005, the data was filtered and analysed.

For question 5, the variables that immediately impact arrival and departure delay were taken. These are ‘CRSDepTime’, ‘CRSArrTime’, ‘DepTime’, ‘ArrTime’, ‘Origin’, and ‘Dest’. Total delay was calculated, and the 6 variables were used to model for total delay. The sample size taken for both Python and R is 200,000. A large sample size results in greater accuracy within the model. For R, the variables ‘Origin’ and ‘Dest’ had to be dropped due to the “Factor has new levels” error that occurs from taking samples. Unfortunately, this results in lower accuracy but the error could not be solved. The only solution found was to take the entire dataset but this would be impossible due to the fact that the data has about 7 million rows. This will result in numerous memory errors and session aborts from R. Hence, the variables were dropped. The models used are logistic regression and gradient boosting in Python and linear regression, ridge regression, lasso regression, and random forests.

4) Data processing, exploration and modelling

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

4.1.1) The best time of the day

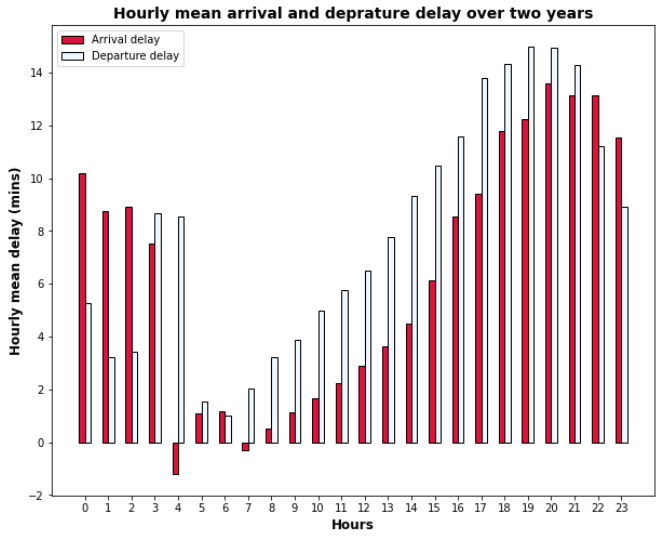

Figure 3: Hourly mean of total delay (Python)

As we can see in the hourly mean arrival and departure delay visualizations from both R and Python, the lowest arrival delay mean is in the morning, at 04:00 and 07:00 hours. The lowest departure delay mean is between at 06:00 hours. The second lowest is at 05:00 hours.

The highest arrival delay mean is in the evening, at 18:00 to 23:00 (6 to 11 PM) which then also continues over into the next day at midnight (12 AM or 00 hours).

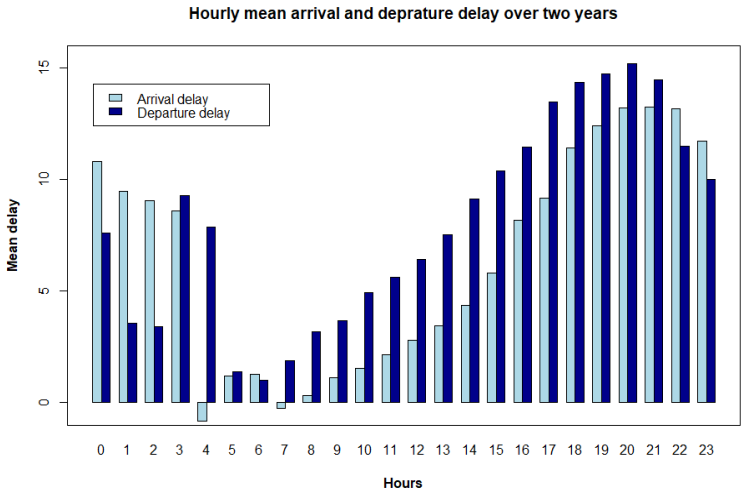

Figure 4: Hourly mean of total delay (R)

This persists till 04:00 hours where it drops into the negatives. The highest departure delay mean is also in the evening, at 18:00 to 23:00 (6 to 11 PM) which then also continues over into the next day at midnight (12 AM or 00 hours).

In general, the time between 14:00 to 22:00 (2 PM to 10 PM) is when a lot of delays occur. There is also an abnormal increase at 03:00 to 04:00 in the early morning. Hence, the best time of the day to fly, in order to minimise delays, is during the early morning, from 05:00 to 06:00. In order to avoid peak delays, a person should avoid the evening hours of 17:00 to 21:00 and early morning hour at 04:00.

4.1.2) The best day of the week

The best day of the week, where total delays are the lowest, is Saturday. The total delay mean is close to ten minutes on Saturday while the second lowest is Tuesday where it is close to 12.5 minutes. The highest total delay mean is on Thursday and Friday with both having more than 17.5 minutes in the total delay mean. The next highest is Monday with just under 17.5 minutes.

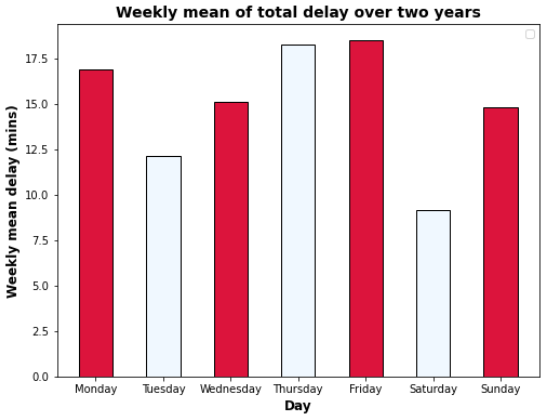

Figure 5: Weekly mean (Python)

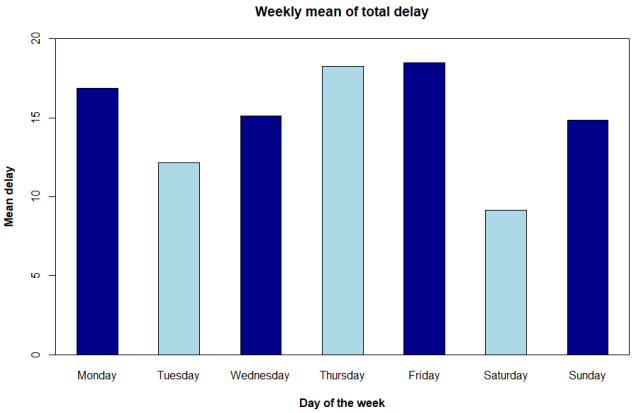

Figure 6: Weekly mean (R)

Hence, the best day of the week is Saturday. To avoid delays, it is best to avoid Thursdays, Fridays, and Mondays.

4.1.3) The best time of the year (month)

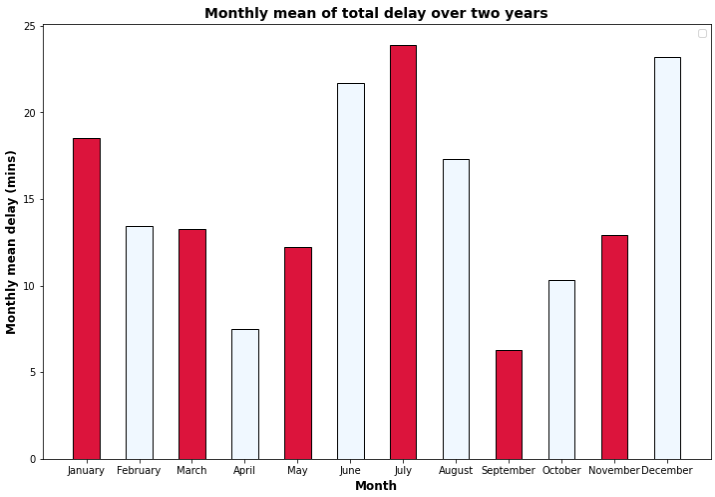

Figure 5: Monthly mean (Python)

The best time of the year to fly, by month, is September which has a mean of about six minutes. The next best month is April which has a mean of about seven minutes.

The highest mean is during July, with December and June coming in 2nd and 3rd place for highest mean. This corresponds to the Summer and Winter break, hence the rise in

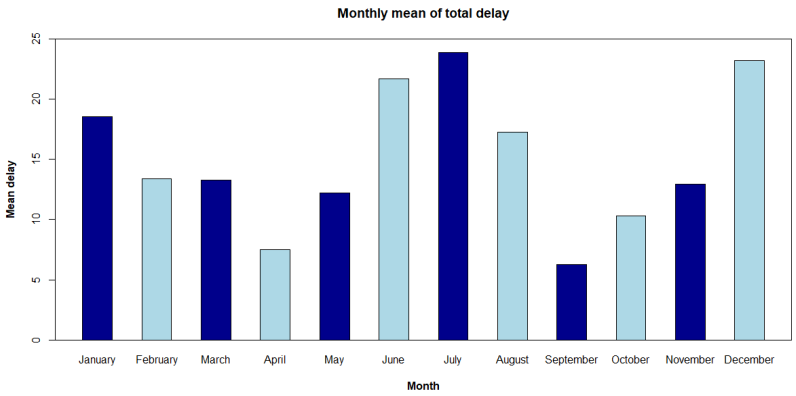

Figure 6: Monthly mean (R)

delays. January also has high delays due to spill over from Winter break and New Year's. It can also include the weather from winter season in December and January. The same can be seen in the R visualization.

4.2) Do older planes suffer more delays?

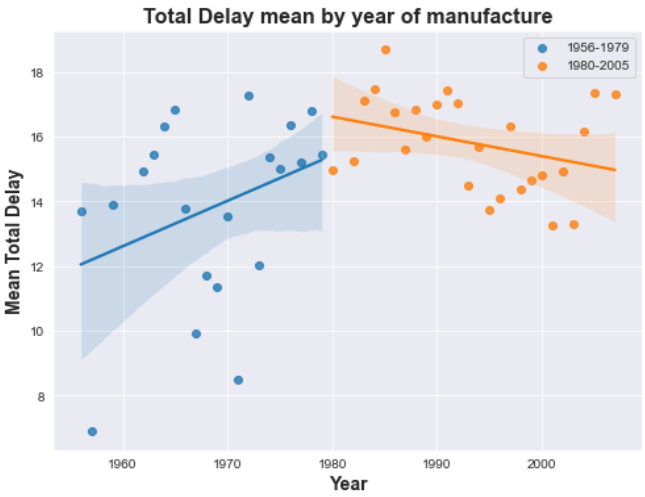

Figure 9: Regression plot of total delay against year of manufacture (Python)

There are two regression lines in each plot. This is done due to the sampling difference mentioned in the data pre-processing. It is helpful to split the data and plot them in two sections within the same plot.

The data is much more scattered in pre-1980s as opposed to post-1980s where there is much more data that results in a more gathered plot. Likewise, the confidence interval is much wider for pre-1980s as compared to post-1980s. Since there is a sampling issue with pre-1980s, we cannot reasonably apply the conclusion for all years. Hence, 1980 onwards, data points and analysis show that older planes did suffer more delays. We can see that there is a slow downward trend with much more scatter appearing below the trend line post-1990s and that there is much

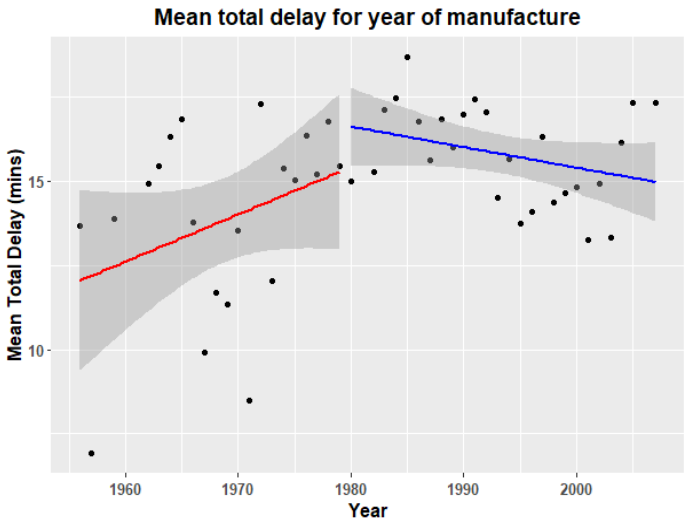

Figure 10: Regression plot of total delay against year of manufacture (R)

more scatter above the trend line during the 1980s. Meanwhile, pre-1980, the delays were increasing as seen by the upward sloping regression line.

4.3) How does the number of people (number of flights) flying between different locations change over time?

In both visualizations, we can see that there is quite a lot of change for most of the connections. The largest change is between DTW - TYS (Detroit, Michigan – Knoxville, Tennessee) where more than 90% of total flights were in 2004 which dipped to less than 5% of total flights in 2005. The second largest change is between LAS – PVD (Las Vegas, Nevada – Warwick, Rhode Island) where it had a little over 10% of total flights in 2004 which then increased to more than 80%. DTW – SBN (Detroit, Michigan – South Bend, Indiana) and LGA – STT (Queens, New York – St. Thomas, US Virgin Islands) have about the same amount of change, percentage wise. Both had just below 40% in 2004 which increased to more than 60% in 2005. AKN – DLG (King Salmon, Alaska – Dillingham, Alaska) had the opposite change with 60% in 2004 dropping to 40% in 2005.

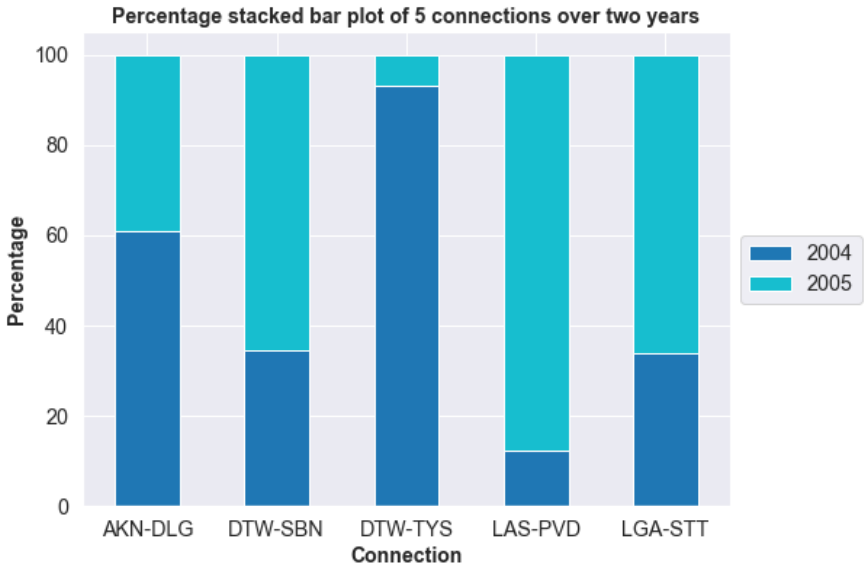

Figure 7: Top 5 connections by total flights (Python)

The R visualization is quite interesting, almost resembling a step graph. The largest change is, of course, between MSY – SLC (New Orleans, Louisiana – Salt Lake, Utah) where the percentage change is 100% which means that the flights in 2005 are double of those in 2004. The second largest is between GSO – TPA (Greensboro, North Carolina – Port Alsworth, Alaska) where there was an increase from 25% to 75% of total flights. There is a similar increase between GRB – MQT (Green Bay, Wisconsin – Gwinn, Michigan) where it increased from about 30% to 70% from 2004 to 2005. The connection between AKN – ANC (King Salmon, Alaska – Anchorage, Alaska) had almost no change at all with the percentage increasing from just below 50% to just a little over 50% from 2004 to 2005. A similar change

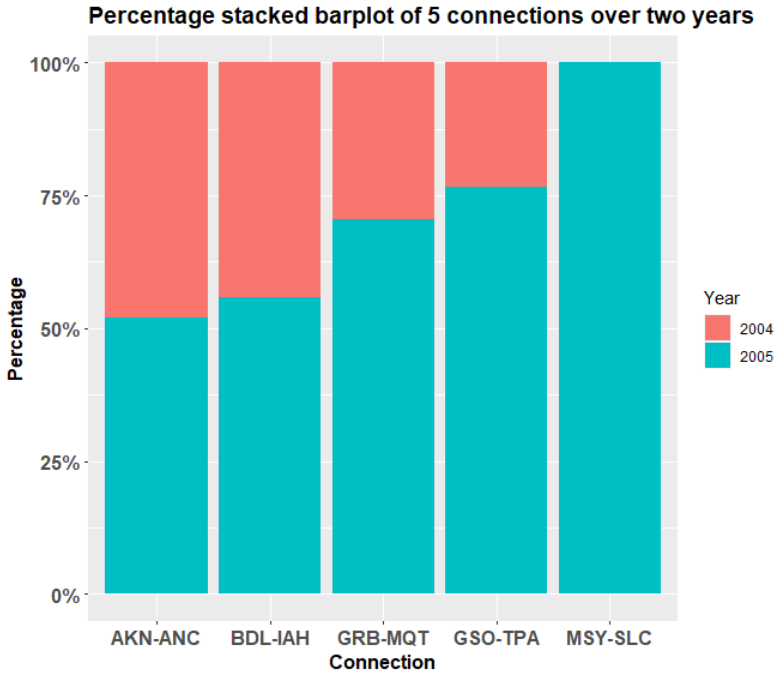

Figure 8: Top 5 connections by total flights (R)

is noted in BDL – IAH (Windsor Locks, Connecticut – Houston, Texas) with the increase being from about 45% to 55%.

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

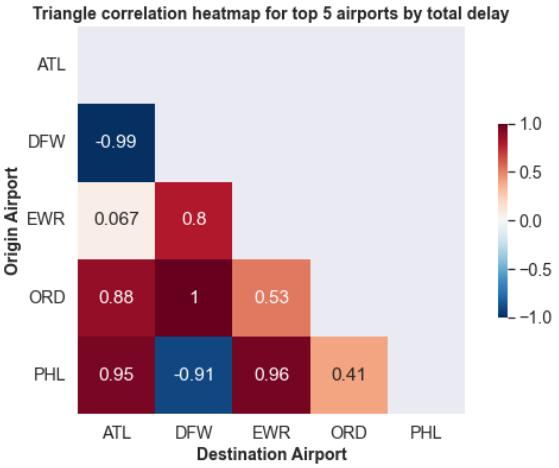

Figure 13: Triangle correlation heatmap of top 5 airports (Python)

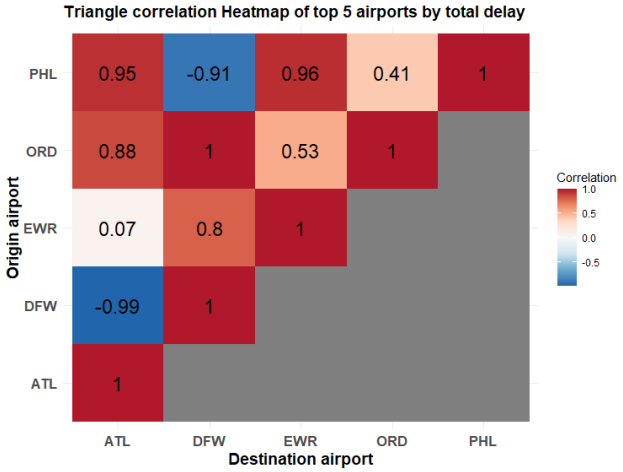

Figure 14: Triangle correlation heatmap of top 5 airports (R)

It is possible to detect cascading failures as delays. The visualizations from Python and R both show the same information. We can that there are indeed delays which cascade from one airport into another. There is perfect correlation between ORD (O'Hare airport, Chicago) and DFW (Dallas/Fort Worth airport, Dallas-Fort Worth) which means that the delays in

ORD constantly cascade to DFW and result in delays in DFW. There are other similarly high correlations between other airports. PHL (Philadelphia airport, Philadelphia) and ATL (Hartsfield-Jackson airport, Atlanta) have a delay correlation of 0.95 while PHL has the pretty much the same correlation with EWR (Newark Liberty airport, Newark) that has a correlation of 0.96. ORD and ATL have a correlation of 0.88 while EWR and DFW have a correlation of 0.8. The high positive correlations imply that delays in one airport, like PHL, do translate to or be associated with delays in the other airport, like ATL. This could be an indication that cascading failures are occurring between the airports. There is also much lower correlation such as 0.53 between ORD and EWR as well as 0.41 between PHL and ORD. This means that there is a moderately positive relationship in delays between the airports but does not necessarily mean that cascading failures are occurring. It is also to be noted that there is a correlation of 0.067, about 0, between EWR and ATL. This means that the delays in EWR do not affect the delays in ATL. There are also negative correlations between DFW and ATL, of -0.99, as well as between PHL and DFW, of -0.91. This means delays in DFW and PHL do not translate to delays in ATL and DFW, respectively.

As we saw, we can detect cascading failures as delays that start from one airport and lead to delays in other airports. Namely, there are indications of cascading failures between ORD – DFW, PHL – EWR, PHL – ATL, ORD – ATL, and EWR – DFW.

4.5) Use the available variables to construct a model that predicts delays

Figure 15: Machine learning model (Python)

The train-test split is 80-20. A large sample was used in order to improves accuracy. We can see that we have an AUC score of 0.89 for the gradient boosting while the logistic regression has an AUC of 0.71. An AUC score tells us how well a model can predict classes correctly [4][5][6]. An AUC score of 0.71 means that the logistic regression model can distinguish and predict 71% of the classes correctly. This is already a good score.

Gradient boosting has a score 0.89 which means that it can predict 89% of the classes correctly. This is a high score and is close to being very high since it is just 1% below 90%. We can improve on the model by adding more variables that contribute to the delay [4]. We can also change the train-test ratio and see if it can improve the model.

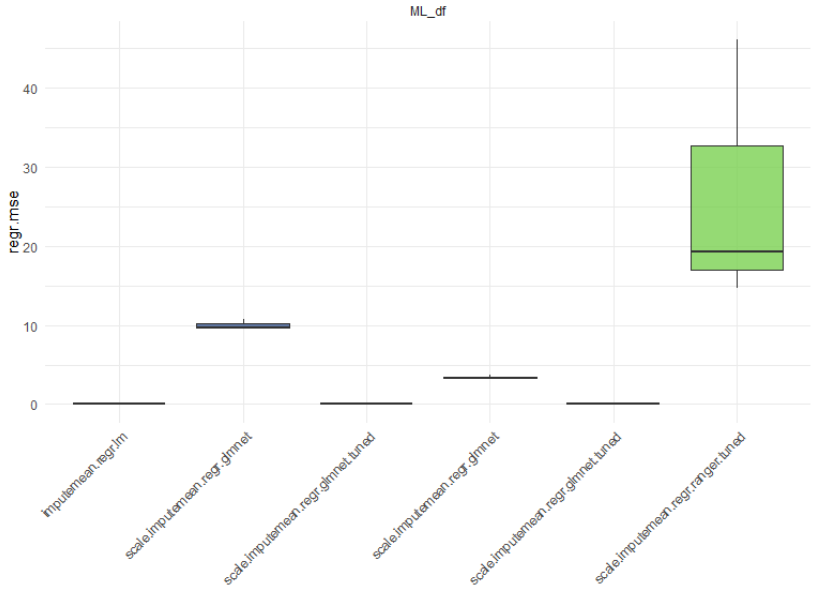

Figure 16: MSE values of regression models (R)

On the left is the graph visualizing the various mean squared error (MSE) values for the regression models. The values for each model are given in the table below. As we can see, the MSE values vary quite a bit. The random forest model has the highest MSE with the most range as well. Its mean MSE value is 26.6. The second highest is the ridge regression without any tuning. The third highest is the lasso regression without any tuning. The lowest is the linear regression. A MSE value of zero would imply a perfect model [7]. We cannot conclude this with the linear regression as such a small value when variables like 'Origin' and 'Dest' were dropped. It also threw a warning message that the "prediction from a rank deficient fit may be misleading". This means that the test data did not work well with the regression [8]. The ridge and lasso regressions without tuning are more believable but these are without tuning. Their penalty terms, also called lambdas, have not been calculated when benchmarking [9]. We can with the penalty terms that their MSE values falls drastically. We can improve upon the model by including more variables, especially origin and destination.

5) Conclusions

Overall, we were able to analyse when delays occur, changes in flight connections, cascading delays etc. There can be much more flight analysis done with this data. The entirety of the code, both in Python and R, took around 30 minutes to run and execute which is quite fast considering the samples taken in the machine learning models.

6) References

[1] The World of Air Transport in 2019. (n.d.). Retrieved March 25, 2023, from https://www.icao.int/annual-report-2019/Pages/the-world-of-air-transport-in-2019.aspx

[2] Harvard Dataverse. (2008, October 6). *Data expo 2009: Airline on Time Data*. Harvard Dataverse. Retrieved February 27, 2023, from https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi%3A10.7910%2FDVN%2FHG7NV7

[3] Mears, K., Montelpare, W. J., Read, E., McComber, T., Mahar, A., & Ritchie, K. (n.d.). *Working with missing data*. Applied Statistics in Healthcare Research. Retrieved April 3, 2023, from https://pressbooks.library.upei.ca/montelpare/chapter/working-with-missing-data/

[4] Allwright, S. (2022, August 23). *What is a good AUC score? (simply explained)*. Stephen Allwright. Retrieved April 2, 2023, from https://stephenallwright.com/good-auc-score/

[5] Google. (n.d.). *Classification: Roc curve and AUC | machine learning | google developers*. Google. Retrieved April 1, 2023, from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[6] Bhandari, A. (2023, March 28). *Guide to AUC ROC curve in machine learning : What is specificity?* Analytics Vidhya. Retrieved March 25, 2023, from https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

[7] Allwright, S. (2022, December 6). What is a good MSE value? (simply explained). Stephen Allwright. Retrieved April 2, 2023, from https://stephenallwright.com/good-mse-value/

[8] *Fix R warning message: Prediction from a rank-deficient fit may be misleading*. ProgrammingR. (n.d.). Retrieved April 1, 2023, from https://www.programmingr.com/r-error-messages/prediction-from-a-rank-deficient-fit-may-be-misleading/

[9] *Ridge and lasso regression*. Andrea Perlato. (n.d.). Retrieved April 2, 2023, from https://www.andreaperlato.com/theorypost/ridge-and-lasso-regression/