UNIVERSITY OF LONDON

THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

ST2195-Programming for Data Science

Analysis Report-R and Python

Student ID-210458262

Page count-10 pages excluding cover page and table of contents

Table of Contents

01.Introduction

This report is based on planes that have been flying over different locations overtime and the Departure delays that it has faced when departing from the Origin and Arrival delays when arriving at the Destination airports within the United States of America(USA).

The “2006” and “2007” year datasets obtained from the Harvard Website(https://doi.org/10.7910/DVN/HG7NV7) can be seen to be the latest highest amount of observations included datasets that are available. Therefore, those were chosen to be investigated in both R and Python programming languages. Moreover, other data files on Airports and plane data were also used when answering certain questions.

This report includes about, Section 01-The data cleaning process. Section 02- The best time of day, day of the week, and time of year to fly to minimise delays. Section 03-Whether older planes suffer more delays. Section 04-The number of people flying between different locations changing over the time. Section 05-Detecting cascading failures as delays in one airport create delays in others. Section 06-Constructing a model that predicts delays.

02.Data Cleaning and Pre-Processing

As the columns of both the 2006 and 2007 year datasets provided to us were the same, they were combined to form a single dataset. The other datasets, airports and plane data were left merged for analysis when needed. The duplicated rows from all the datasets were removed initially. Since delays less than 0 means that the plane has departed early, delays equal to 0 means that the plane has departed on time and delays greater than zero means there has been a delay, these values were also dropped when analyzing the best time of day, week and year. However, it was used when detecting cascading failures. Also, cleaning is done for each analysis separately by dropping null values only for the data columns needed as, dropping missing values initially for the whole dataset will mean loss of data which could have been used to the analysis as the null values were only appearing in a different column leading to useless loss of data. Even though, outliers were identified it was decided not to drop as we assume that it will tend to lose important information on delays. When it comes to modelling we created indicator variables and standardized numerical data when needed.

03. Analysis on the best time of day, day of the week, and time of year to fly to minimise delays

The analysis is done for both Arrival and Departure delays as it can be seen in Figure 13 that the correlation between arrival and departure delay is 0.92 that shows that the delays are relatable. Therefore, two data frames are made for each arrival and departure delay with Month, DayOfWeek, CRSDepTime for each and the respective ArrDelay and DepDelay for cleaning purposes. In order to determine which central tendency to be used in the analysis a boxplot is created for both the delays. Here the outliers will be removed due to visualization purposes.

Using Figure 1 below, since both distribution of the delays shows positively skewed shapes and due to the presence of many outliers the analysis will be done using median as mean will not be appropriate.

Python *R*

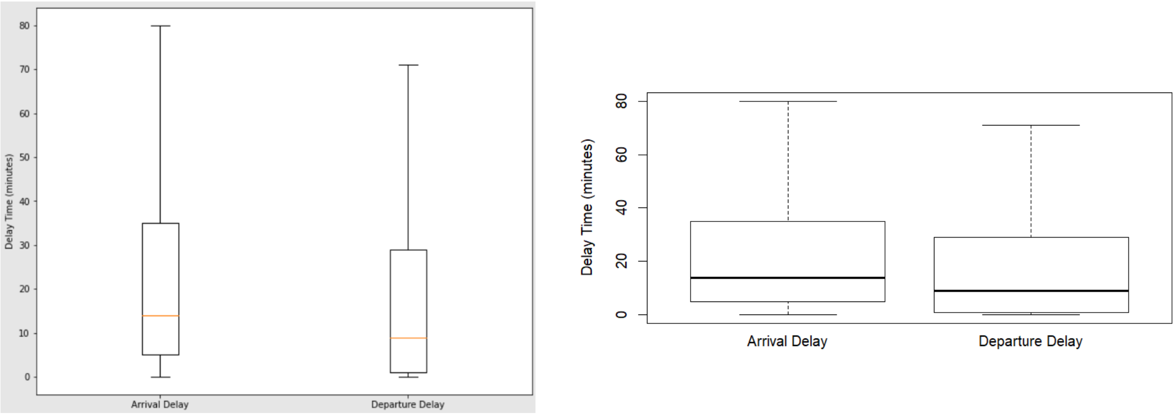


Figure 1. Arrival and Departure Delay Boxplot

For analysis of best time of day, day of week and time of year line graphs will be used as it helps to give a clear picture on how the median arrival and departure delays have fluctuated within the 24 hours of day, 7 days of week and by the 12 months. To get a better picture the total effect on delays is shown as total median delay which is the addition of arrival and departure median delays for each hour, days of week and by month.

3.1 Best Time of Day to Fly to Minimise Delays

Python *R*

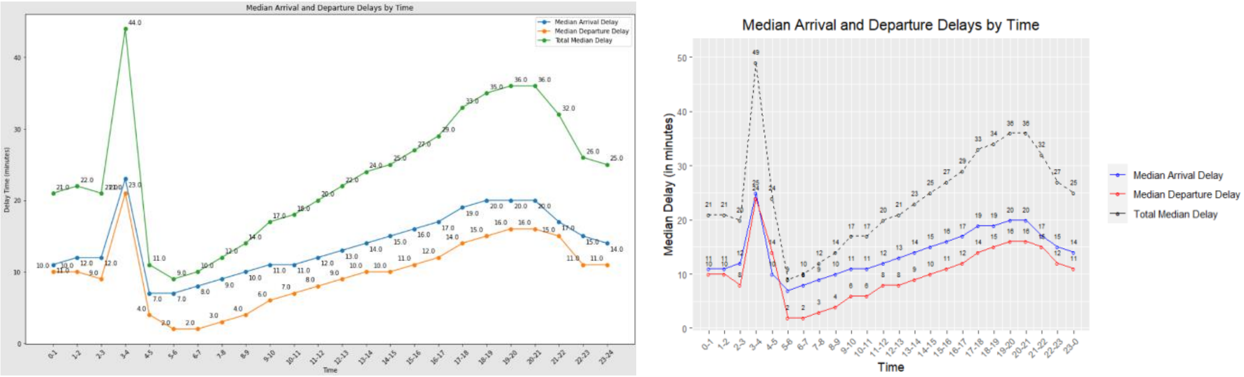


Figure 2. Median Arrival and Departure Delays by Time

Using Figure 2, the best time of the day when the delays were at the lowest is from 05.00 to 06.00 when considering both the delays. Even though, 04.00 to 05.00 and 05.00 to 06.00 has the lowest median arrival delay the median departure delay is higher at 04.00 to 05.00 than 05.00 to 06.00 making it the best. Same goes with 05.00 to 06.00 and 06.00 to 07.00 which has the lowest median departure delay but the arrival delay is higher from 06.00 to 07.00 than 05.00 to 06.00 which makes it the best time. Also, it can be seen that both the delays are high from 03.00 to 04.00 time period.

3.2 Best Day of Week to Fly to Minimise Delays

Python *R*

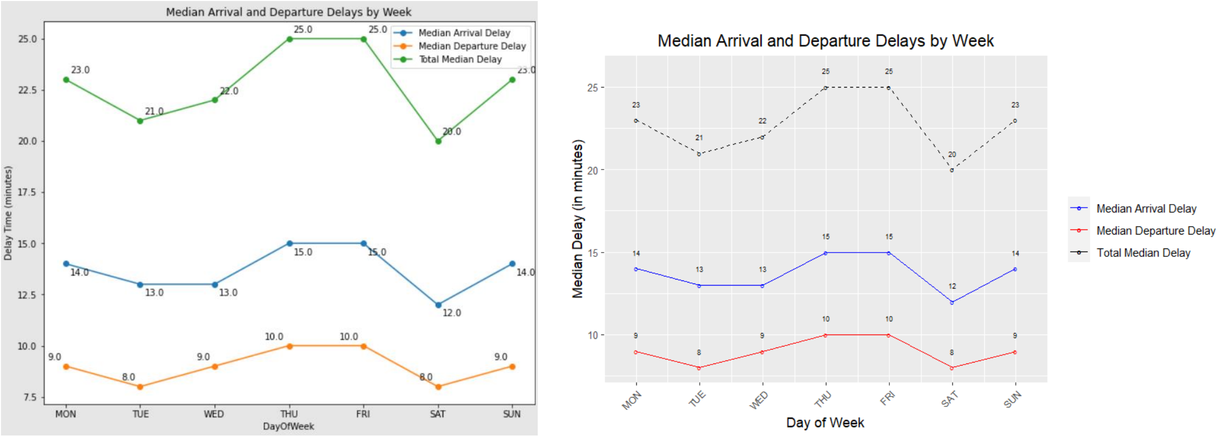


Figure 3. Median Arrival and Departure Delays by Week

Using Figure 3, the best day of the week to fly with lowest median delay is on Saturday with both the delays at the lowest. Even though, both Tuesday and Saturday has the lowest median departure delay it can be seen that the median arrival delay on Tuesday is higher making Saturday the best. Also it can be noticed that Thursday and Friday are both with the highest median arrival and departure delay making it overall highest delayed days.

3.3 Best Time of Year to Fly to Minimise Delays

Using Figure 4, the Best time of week to fly with the lowest median delay can be seen as the month of May. Even though, the months April, May, September and November have the lowest median arrival delay it can be seen that month of May only has the lowest median departure delay across the year making it the best. Also, it can be seen that the month of December had the highest median delay across the year.

Python *R*

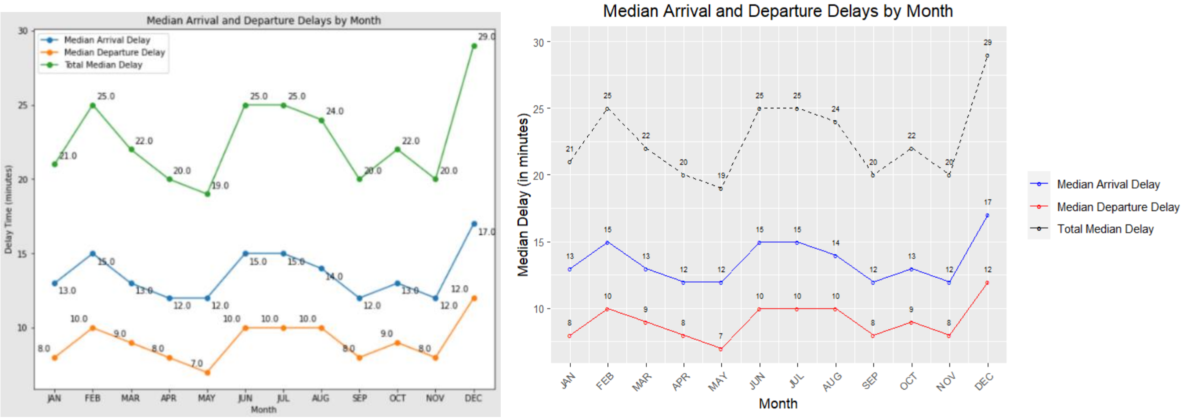


Figure 4. Median Arrival and Departure Delays by Month

04.Analysis on whether older planes suffer more delays

The analysis is done for both arrival and departure delays as well as delay types to see the relationship between plane age and delays. Therefore, the plane data set is merged based on the Tail number and plane age is calculated with the difference of the current year and year of manufacture. Afterwards 3 data frames are made and cleaned where 0 was determined as a null value in delay types as it affects with the mean calculation. When analyzing to see whether there is a relationship between plane age and delays we make use of scatter plots and correlation calculations and a line graph to see the trend of delays over plane age.

Python *R*

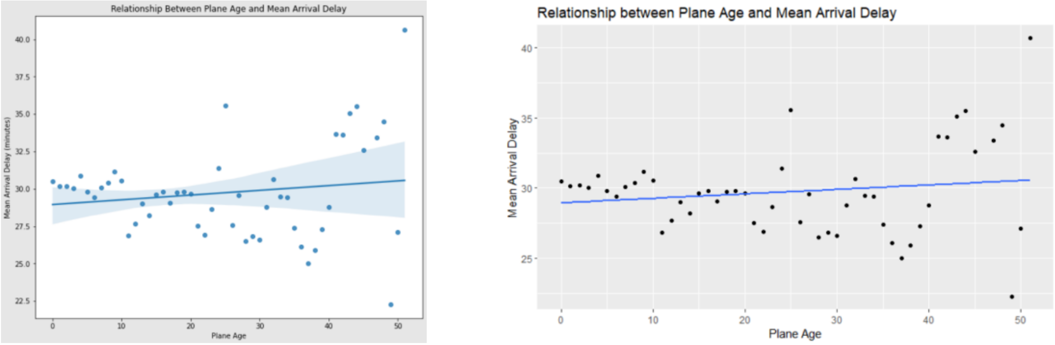


Figure 5 Relationship Between Plane Age and Mean Arrival Delay

Using Figure 5 we can see that the relationship between the plane age and mean arrival delay is a weak positive relationship and this is justified by correlation coefficient 0.152 that the effect of plane age on arrival delays is low.

Python *R*

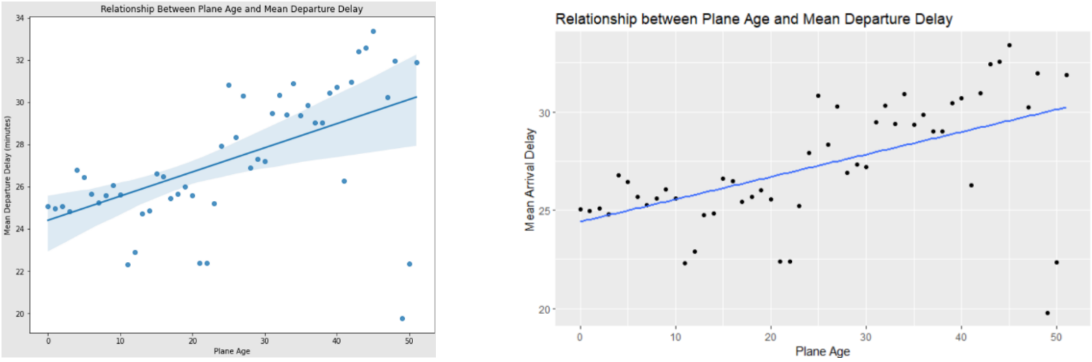


Figure 6 Relationship Between Plane Age and Mean Arrival Delay

Using Figure 6 we can see that the relationship between the plane age and mean departure delay is a moderately strong positive relationship and justified by correlation coefficient 0.549 that there is an effect of plane age on delays. The below line plot in Figure 7 shows us that there are fluctuations in the line as the plane age increases but it doesn’t necessarily show a trend (increasing or decreasing). There is a large rise in mean delay for the plane age within 20 and 30 years but however has a drastic drop and rises at the late 40 s. Therefore, there is not necessarily much information to show that older planes do suffer more delays.

Python *R*

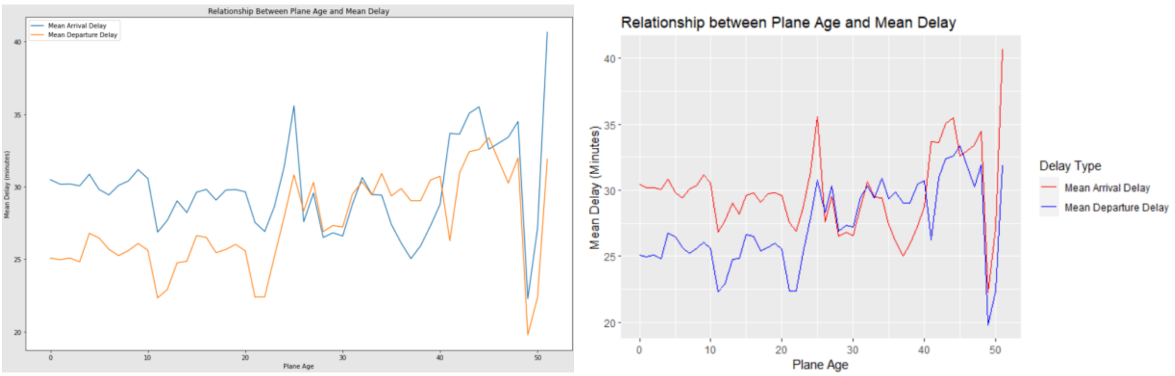


Figure 7 Relationship Between Plane Age and Mean Delay

Figure 8 shows the relationship between plane age and mean delay types. Here, only carrier delays have a moderate positive relationship with a correlation coefficient of 0.528. Meanwhile the other delay types have a negative correlation coefficient showing no relationship at all with plane age; Weather Delay -0.302, NAS Delay -0.269, Security Delay -0.009 and Late Aircraft Delay -0.450. This is because the reason for these delays to happen has nothing to do with plane age.

Python *R*

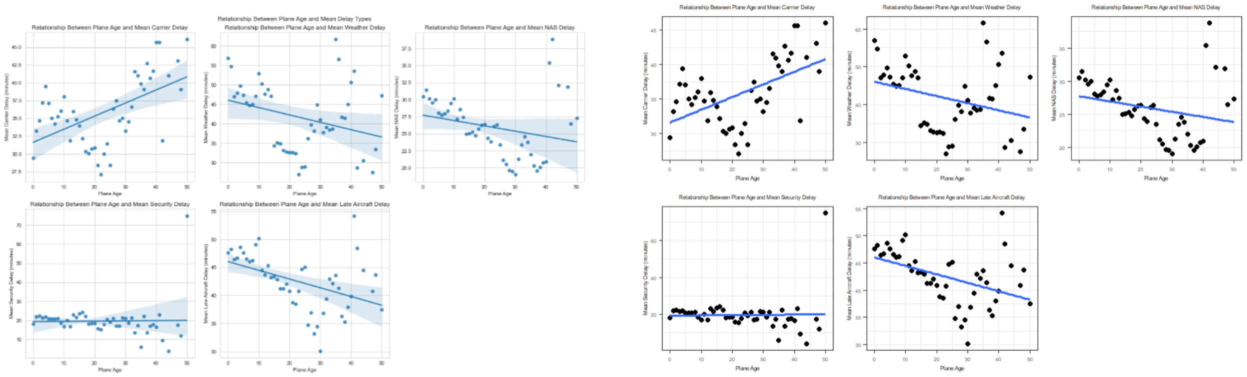
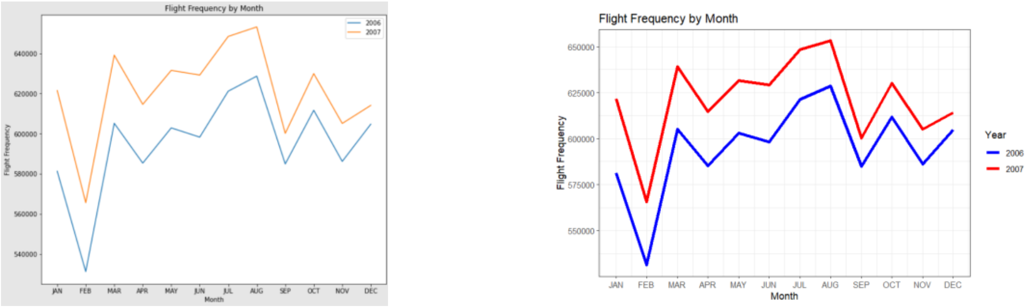


Figure 8 Relationship Between Plane Age and Mean Delay Types

05. Analysis on the number of people flying between different locations changing over time

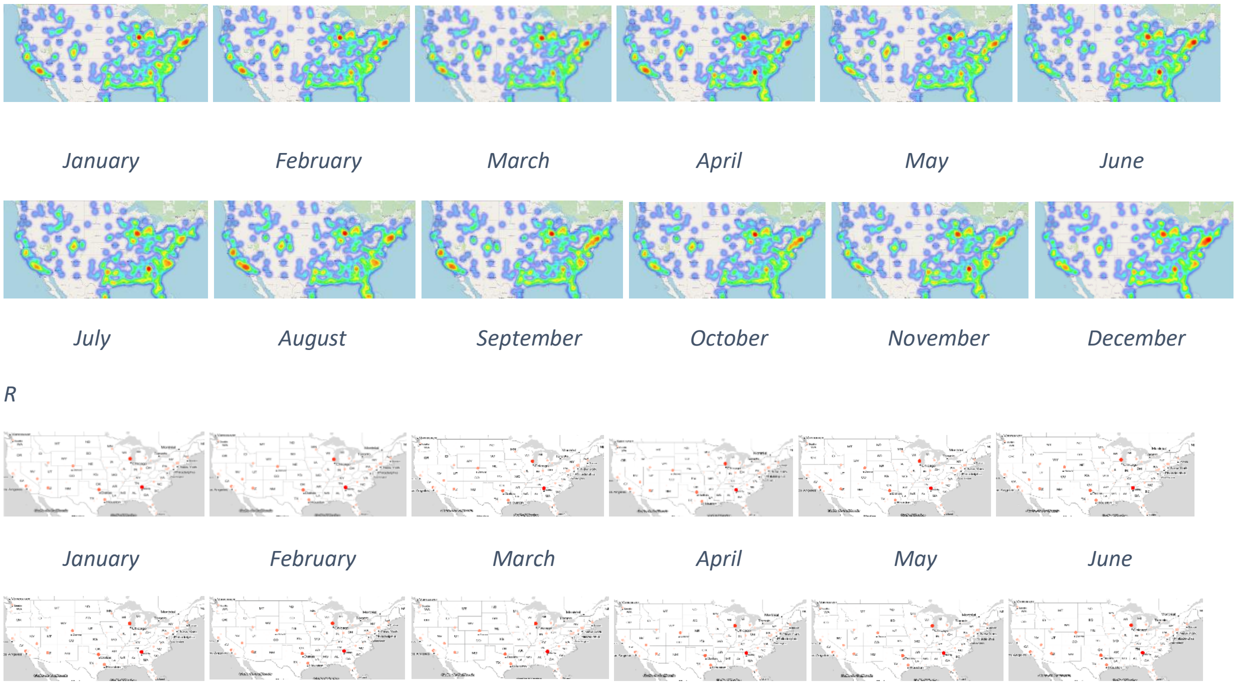
The analysis is done based on frequency destinations for each month. Here the initial merged data set is left merged by the airports dataset and is cleaned after choosing Month, Year, Dest, airport, state , lat and long columns for the analysis. Here we use Line graph as well as Spatio Temporal Heat Map of the USA map to see how people changed their destination through time.


Python

R

Figure 9. Flight Frequency by Month

Figure 9 shows that the trend between 2006 and 2007 is the same. However, the number of flights have increased among the years. But we can see that in February there is a drastic fall this can be because USA faces winter during these periods whereas there is a rising trend from March to August during spring and summer which is when people tend to travel. Since the trend of 2006 and 2007 years are the same in below Figure 10 we only analyse on 2006 frequencies through a heat map which is shown within USA bounds for visualization purposes.


Python

January *February* *March* *April* *May* *June*

July *August* *September* *October* *November* *December*

R

January *February* *March* *April* *May* *June*

July *August* *September* *October* *November* *December*

Figure 10. Spatial Temporal Heat Map for each month for the year 2006

Based on Figure 10, it can be seen that throughout the year based on destination it cannot be said that people frequented in a particular destination the most at a time period. This can be shown from Figure 11 that among the top 3 states the frequency never showed with California, Texas and Illinois overcrossing each other and it can be seen that the frequency rises and falls at the same time of the year.

Python *R*

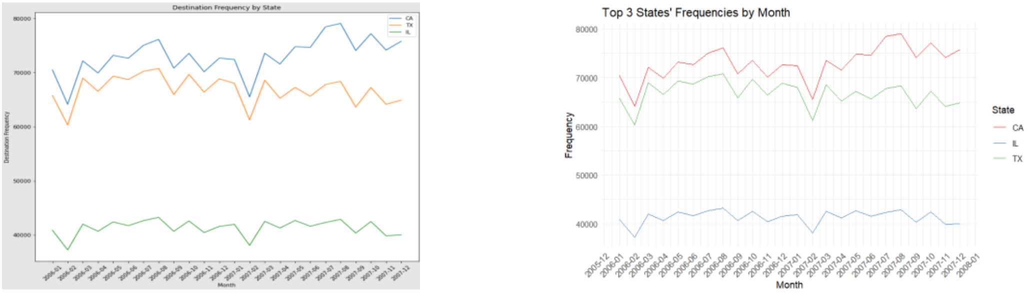


Figure 11. Top 3 States Frequencies by Month

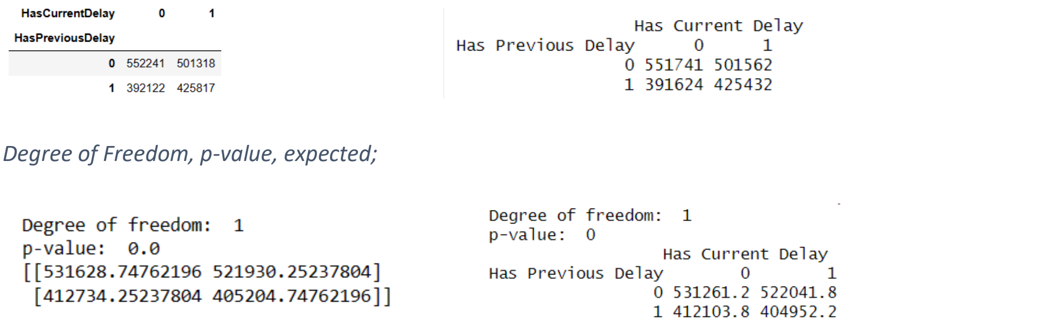
06. Can we detect Cascading failures on whether delays in one airport create delays in others?

For this analysis, the dataset is cleaned with a date/time column with origin, destination and total delays and tail num. In order to check whether the current delay was affected by previous flight delay a new data frame was created by grouping the data by tail number and sorting according to date and time and if the origin equaled to previous flights destination then it will be labelled with previous and current values of the column. However, in the process the 2 resulting data frames of python and R gives different dimensions due to different libraries being available by different languages. Here we use hypothesis testing in order to make a clear analysis.

The Hypothesis test will be defined with H0 saying that there is no association between previous delays and current delays and with H1 that there is an association between previous delays and current delays.

Python *R*

Crosstab:



HasCurrentDelay	0	1
HasPreviousDelay		
0	552241	501318
1	392122	425817

```
                    Has Current Delay
Has Previous Delay      0      1
                 0 551741 501562
                 1 391624 425432
```

Degree of Freedom, p-value, expected;

```
Degree of freedom:  1
p-value:  0.0
[[531628.74762196 521930.25237804]
 [412734.25237804 405204.74762196]]
```

```
Degree of freedom:  1
p-value:  0
                      Has Current Delay
Has Previous Delay        0        1
                 0 531261.2 522041.8
                 1 412103.8 404952.2
```

Figure 12. Hypothesis Crosstab, Degrees of freedom, p-value and the expected

Here the crosstab shows us on whether there was a delay with 1 and no delay with 0 and see the combination of previous and current delays. After performing a Chi squared test to check the significance of the relationship between previous delay and current delay by using p-value as shown in Figure 12 with 5% and 1% significance level it can be concluded that H0 was rejected at both 5% and 1% significance level and that there is strong evidence against H0 showing that there is an association between previous delays and current delays.

07. Constructing model to predict delays

Here we build a model to predict arrival delays. In order to consider which values to be taken when building the model, a correlation plot is used to choose the variables with a higher correlation and is most related with arrival delays. The plane data dataset is left merged in order to consider plane age.

Python *R*

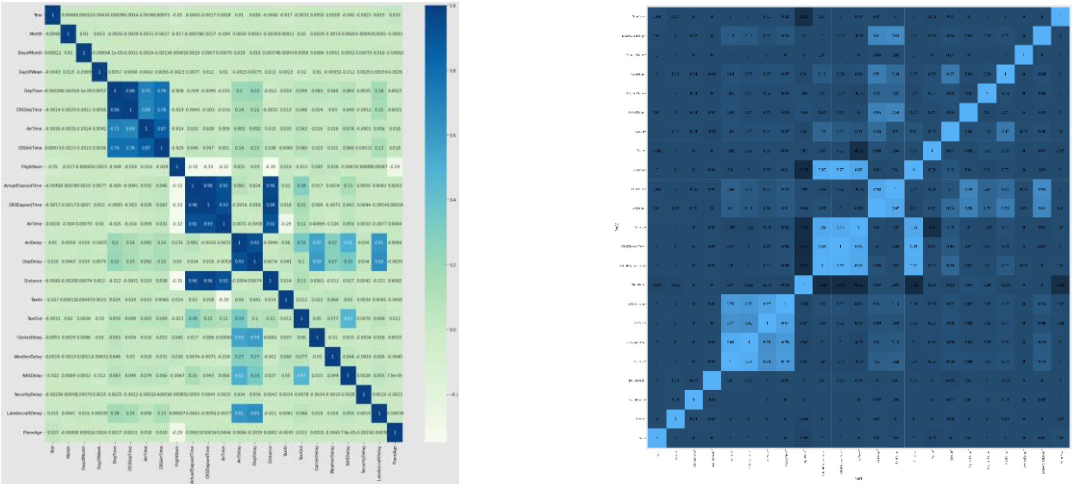


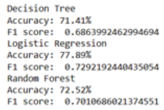
Figure 13. Correlation Heat Map

Even though there are other variables correlated with Arrival Delay we go with Month and Day of Week which will be our categorical variables and departure delay, distance and plane age as numerical variables. Here we build a model for supervised classification in order to predict the delay status, whether the plane will come late or not. Therefore, we add a column with the binary values of 0 and 1 based on the delay status which can help passengers to predict on whether their flight will be delayed or not. Then we build a linear regression model to predict the arrival delays which helps passengers to find out on how long the arrival will be delayed if their delay status becomes 1. Here when data is being trained, test and split we use 70% of data to train and 30% to test the data.

7.1 Supervised Classification

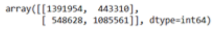
Firstly, when preparing data for classification we drop the arrival delay column as it is not needed and check whether the delay status is balanced and not biased. Here it can be seen that the delay status is in 52:48 ratio showing us that it is approximately balanced. Also, the numerical data will be standardized and the categorical data be given indicator variables. Then we split data, train and test for 3 types of models to check which one has the best accuracy. Here we use Decision Trees, Logistic Regression and Random Forest.

Python



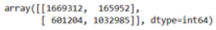
```
Decision Tree
Accuracy: 71.41%
F1 score:  0.6863992462994694
Logistic Regression
Accuracy: 77.89%
F1 score:  0.7292192440435054
Random Forest
Accuracy: 72.52%
F1 score:  0.7010686021374551
```

Decision Tree Confusion matrix



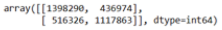
```
array([[1391954,  443310],
       [ 548628, 1085561]], dtype=int64)
```

Logistic Regression Confusion matrix



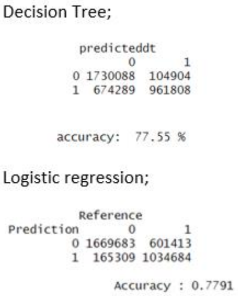
```
array([[1669312,  165952],
       [ 601204, 1032985]], dtype=int64)
```

Random Forest Confusion Matrix



```
array([[1398290,  436974],
       [ 516326, 1117863]], dtype=int64)
```

R



Decision Tree;

```
       predicteddt
              0       1
     0 1730088  104904
     1  674289  961808


    accuracy:  77.55 %
```

Logistic regression;

```
          Reference
Prediction       0       1
         0 1669683  601413
         1  165309 1034684

            Accuracy : 0.7791
```

Figure 13. Results of the models

We can see that logistic regression has the highest accuracy out of all models. In Figure 14, ROC curve for logistic regression shows that it has 0.84 area under curve which is closer to ideal amount of 1. However, methods like cross validation can help to increase model's accuracy further from 77.9%. Since Random Forest's complexity the model could not be built in the R language due to memory problems. Therefore, selecting the model will not always depend on its accuracy but also there will be concerns on model's simplicity.

Python *R*

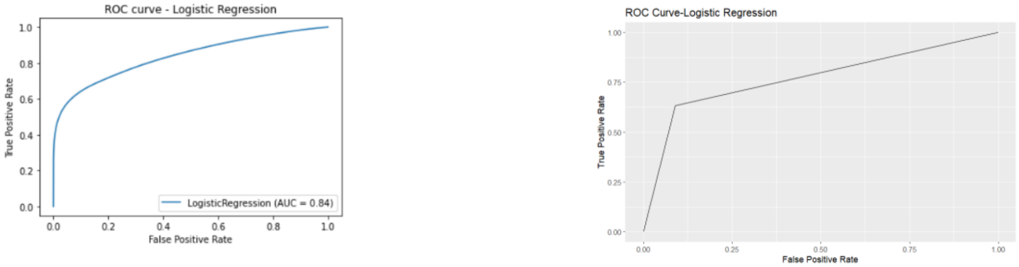
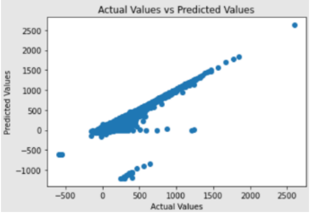


Figure 14. ROC curve- Logistic Regression

7.2 Multiple Linear Regression

The same dataset is used to predict arrival delays. Here, we only create indicator functions on categorical variables and use numerical variables without standardizing. When the model is fit to linear regression and the values are predicted the scatter plots in Figure 15 shows its linearity on how similar the predicted values are to its actual values. Figure 16, shows the models accuracy with an r-squared value of 0.856 for both the languages showing how well the model built fit into the train test. Also, the model score is shown as 0.854 showing that there was 85% accuracy of prediction.

Python



R

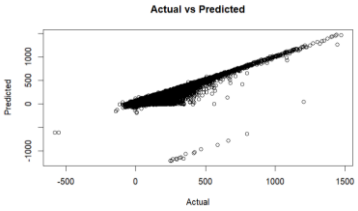
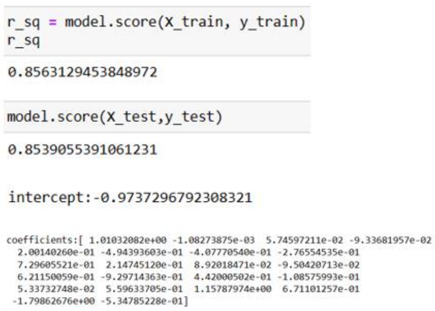


Figure 15. Actual and Predicted Values

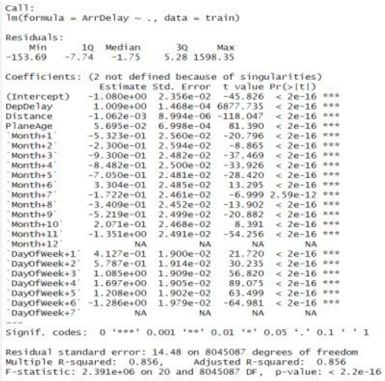
By using Figure 16 we can show the model used as an equation (based on values from python's model); ***Arrival Delay= -0.9737+ 1.010 DepDelay- 1.083*10^-3 Distance+ 5.746*10^-2 PlaneAge-9.337*10^-2 Month1+ 2.001*10^-1 Month2- 4.944*10^-1 Month3+ -4.078*10^-1 Month4-2.766*10^-1 Month 5+7.296*10^-1 Month6+2.148*10^-1 Month7 +8.920*10^-2 Month8-9.504*10^-2 Month9+6.212*10^-1 Month10-9.297*10^-1 Month11+ 4.420*10^-1. Month12-1.086*10^-1 DayOfWeek1+5.337*10^-2 DayOfWeek2+ 5.596*10^-1 DayOfWeek3+1.158 DayOfWeek4+ 6.711*10^-1 DayOfWeek5-1.799 DayOfWeek6- 5.348*10^-1 DayOfWeek7***

Python



```
r_sq = model.score(X_train, y_train)
r_sq

0.8563129453848972

model.score(X_test,y_test)

0.8539055391061231

intercept:-0.9737296792308321

coefficients:[ 1.01032082e+00 -1.08273875e-03  5.74597211e-02 -9.33681957e-02
  2.00140260e-01 -4.94393603e-01 -4.07770540e-01 -2.76554535e-01
  7.29605521e-01  2.14745120e-01  8.92018471e-02 -9.50420713e-02
  6.21150059e-01 -9.29714363e-01  4.42000502e-01 -1.08575993e-01
  5.33732748e-02  5.59633705e-01  1.15787974e+00  6.71101257e-01
 -1.79862676e+00 -5.34785228e-01]
```

R

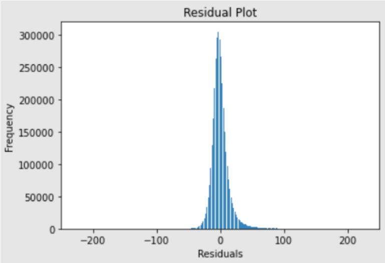


```
Call:
lm(formula = ArrDelay ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-153.69   -7.74   -1.75    5.28 1598.35 

Coefficients: (2 not defined because of singularities)
              Estimate Std. Error  t value Pr(>|t|)    
(Intercept)  -1.080e+00  2.356e-02  -45.826  < 2e-16 ***
DepDelay      1.009e+00  1.468e-04 6877.735  < 2e-16 ***
Distance     -1.062e-03  8.994e-06 -118.047  < 2e-16 ***
PlaneAge      5.695e-02  6.998e-04   81.390  < 2e-16 ***
`Month+1`    -5.323e-01  2.560e-02  -20.796  < 2e-16 ***
`Month+2`    -2.300e-01  2.594e-02   -8.865  < 2e-16 ***
`Month+3`    -9.300e-01  2.482e-02  -37.469  < 2e-16 ***
`Month+4`    -8.482e-01  2.500e-02  -33.926  < 2e-16 ***
`Month+5`    -7.050e-01  2.481e-02  -28.420  < 2e-16 ***
`Month+6`     3.304e-01  2.485e-02   13.295  < 2e-16 ***
`Month+7`    -1.722e-01  2.461e-02   -6.999 2.59e-12 ***
`Month+8`    -3.409e-01  2.452e-02  -13.902  < 2e-16 ***
`Month+9`    -5.219e-01  2.499e-02  -20.882  < 2e-16 ***
`Month+10`    2.071e-01  2.468e-02    8.391  < 2e-16 ***
`Month+11`   -1.351e+00  2.491e-02  -54.256  < 2e-16 ***
`Month+12`           NA         NA       NA       NA    
`DayOfWeek+1` 4.127e-01  1.900e-02   21.720  < 2e-16 ***
`DayOfWeek+2` 5.787e-01  1.914e-02   30.235  < 2e-16 ***
`DayOfWeek+3` 1.085e+00  1.909e-02   56.820  < 2e-16 ***
`DayOfWeek+4` 1.697e+00  1.905e-02   89.075  < 2e-16 ***
`DayOfWeek+5` 1.208e+00  1.902e-02   63.499  < 2e-16 ***
`DayOfWeek+6` -1.286e+00 1.979e-02  -64.981  < 2e-16 ***
`DayOfWeek+7`        NA         NA       NA       NA    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.48 on 8045087 degrees of freedom
Multiple R-squared:  0.856,	Adjusted R-squared:  0.856 
F-statistic: 2.391e+06 on 20 and 8045087 DF,  p-value: < 2.2e-16
```

Figure 16. Model scores, intercept and the coefficients

Moreover, the histogram of the residual plot shown in Figure 17 shows that it is symmetric and is bell-shaped showing that the errors are approximately normally distributed. These results convince that the model we obtained is the best multiple regression model.

Python



R

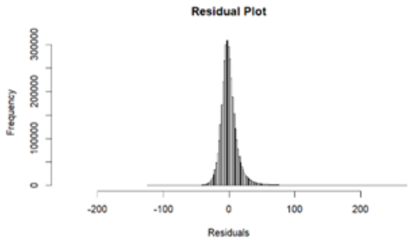


Figure 17. Residual plot