UNIVERSITY OF LONDON

LSE THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

Flight Delay Analysis

Subject Code: ST2195

Subject: Programming for Data Science

UOL Student ID: 210494123

Page Count: 10 (excluding cover page and table of contents)

Table Of Contents

INTRODUCTION & EXECUTIVE SUMMARY

- This report delves into a critical analysis of past flight data regarding flights across the US. The 2006 and 2007 datasets obtained from the Harvard website were selected for analysis as these datasets includes the most number of observations all datasets compared; a large amount of data is required to create a well-functioning machine learning model. Other datasets were also used in addition to the main datasets e.g.csv files for airport and plane-data.

  The report is split into 6 key areas as follows,

- The data cleaning process carried out in order to ensure high quality of information was preserved for investigative analysis.
- Analyze the best time of day, best time of the year and best day of the week to fly in order to minimize delays.
- Investigate if the age of the plane has an impact on the occurrence of delays.
- Analyze how the number of people flying between different locations change over time.
- Investigate if cascading delays in one airport create delays in the following destination airport.
- Use available variables in our dataset to build a model that can predict delays.

- Visualizations created using R Programming and Python was used to support arguments under each key analysis. Also when analyzing delays both mean “ArrDelay” and “DepDelay” were added together and divided by two giving a new average delay value, this method was used across almost all key areas and the procedure is explained inn each key area.Also throughout the analysis for delays we filter both “ArrDelay” >0 and “DepDelay”>0 as delays less than zero are not delays , they are either planes early or on time. Figures and tables obtained from python and R have both been included in all most key areas of interest.

DATA CLEANING PROCESS

- The data cleaning process is done to ensure quality of information used in satisfactory and well preserved for analysis.The 2006 and 2007 datasets were merged together for the cleaning process. The combined dataset has over 14 million observations spread over 29 columns, but one these columns ‘CancellationCode’ was dropped because 98% of the data in the column was missing , the decision to remove the ‘CancellationCode’ was done instead of simply dropping the rows with null values as if drop rows we would be dropping 98% percent of our dataset therefore it is better to drop the column. In addition all duplicate rows were also identified and removed.
- After that we analyze the data to identify any outliers or erroneous values entered that does not make sense. It was discovered that some columns such as ‘Deptime’ & ‘Arrtime’ had entries that exceeded the 24 hour datetime format , these entries were filtered out from the dataset. The cleaned dataset was then saved and used for all analysis.

WHAT IS THE BEST TIME OF DAY, DAY OF WEEK AND TIME OF YEAR TO FLY IN ORDER TO MINIMIZE DELAYS?

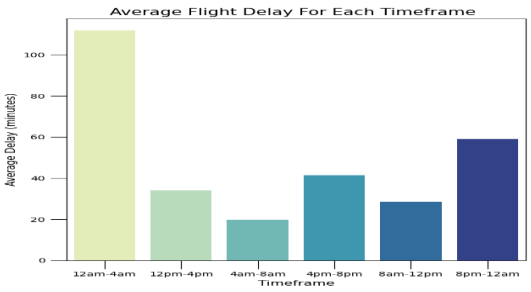
- In order to analyze the best time of day to fly in order to minimize flight delay, 'DepTime' was used to bin into 6 timeframes of 4 hours each , we then group the dataset by 'Timeframe' and obtain the mean 'ArrDelay' and 'DepDelay' for each timeframe and we obtain a new average by getting an average of the two mean delays and recognizing this in a new column 'Total_Delay. We then plot the delay under each timeframe in a bar plot , we see both table values in obtained in R and python below as well,

Timeframe	DepDelay	ArrDelay	Total_Delay
12am-4am	113.881073	109.972640	111.926856
12pm-4pm	32.871590	35.304224	34.087907
4am-8am	17.742700	22.090239	19.916470
4pm-8pm	40.051266	42.720095	41.385680
8am-12pm	27.194735	30.119577	28.657156
8pm-12am	59.016800	58.964379	58.990589

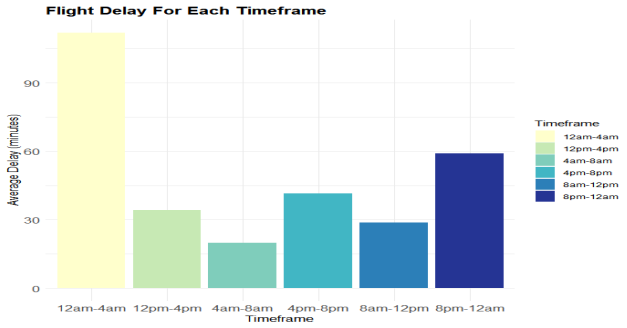
Table: Timeframe (Python)

Timeframe	DepDelay	ArrDelay	TotalDelay
12am-4am	113.88107	109.97264	111.92686
12pm-4pm	32.87159	35.30422	34.08791
4am-8am	17.74270	22.09024	19.91647
4pm-8pm	40.05127	42.72009	41.38568
8am-12pm	27.19473	30.11958	28.65716
8pm-12am	59.01680	58.96438	58.99059

Table: Timeframe (R)



Bar plot: Python



Bar plot: R

- The 4am-8am timeslot has the lowest delay in comparison to other timeframes and therefore is the best time of the day to fly in order to minimize flight delay.

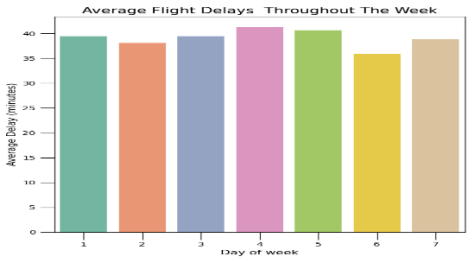
- To analyze the best day of the week to fly we group the dataset by 'DayOfWeek' and obtain the mean arrival and departure delay for each day and then we get the average of the two delays and recognize it in a new column. Visualizations display the data as mentioned, we see both table values in obtained in R and python below as well,

DayOfWeek	DepDelay	ArrDelay	Average Delay
6	35.794975	36.042204	35.918589
2	36.817212	39.325610	38.071411
7	38.071518	39.750057	38.910787
3	38.001987	40.901610	39.451799
1	38.455075	40.469641	39.462358
5	39.441250	41.852456	40.646853
4	39.624418	43.036872	41.330645

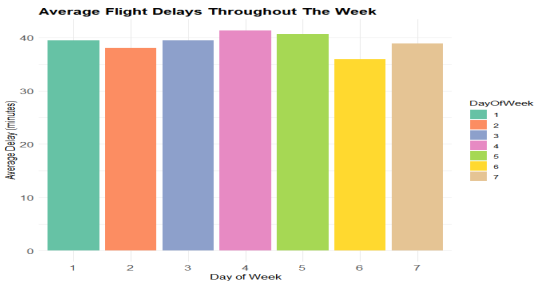
Table: Day of the week (Python)

DayOfWeek	DepDelay	ArrDelay	AverageDelay
1	38.45508	40.46964	39.46236
2	36.81721	39.32561	38.07141
3	38.00199	40.90161	39.45180
4	39.62442	43.03687	41.33064
5	39.44125	41.85246	40.64685
6	35.79498	36.04220	35.91859
7	38.07152	39.75006	38.91079

Table : Day of the week (R)



Bar plot: Python



Bar plot: R

- We see Saturday has the lowest average delay all when compared to all other days of the week. Therefore, Saturday would be the best to travel in order to minimize flight delay.

  - The best time of the year was analyzed in two ways , first we analyzed the best time of year to fly in terms of the month and second we analyzed the best time of the year to fly by quarter of the year. In order to analyze the best time of the year to fly by month we grouped the dataset by 'Month' and obtained the mean arrival and departure delay for each month and then we created a new column with average of the mean of both delays. The barplot and tables provided below should make the process easier to understand. We see both table values in obtained in R and python below as well,

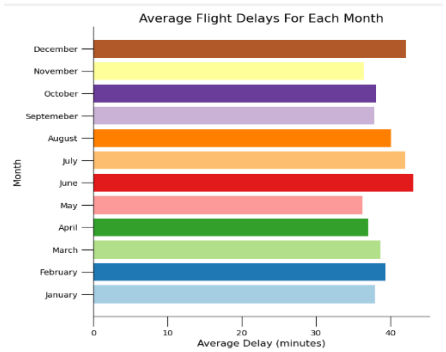
Month	DepDelay	ArrDelay	Total Average Delay
1	36.903217	38.858856	37.881037
2	38.184256	40.488415	39.336335
3	37.778467	39.527578	38.653023
4	36.140453	37.824348	36.982400
5	35.182146	37.284722	36.233434
6	41.534677	44.571320	43.052999
7	40.889315	43.114687	42.002001
8	39.019221	41.198266	40.108744
9	36.599327	39.123740	37.861533
10	36.506594	39.556295	38.031444
11	35.615042	37.174995	36.395019
12	40.880871	43.185546	42.033209

Table: Month Python

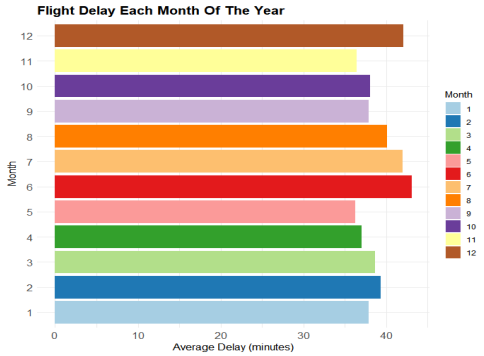
Month	DepDelay	ArrDelay	TotalAverageDelay
1	36.90322	38.85886	37.88104
2	38.18426	40.48841	39.33634
3	37.77847	39.52758	38.65302
4	36.14045	37.82435	36.98240
5	35.18215	37.28472	36.23343
6	41.53468	44.57132	43.05300
7	40.88931	43.11469	42.00200
8	39.01922	41.19827	40.10874
9	36.59933	39.12374	37.86153
10	36.50659	39.55629	38.03144
11	35.61504	37.17500	36.39502
12	40.88087	43.18555	42.03321

Table: Month R

- The best of month of the year of fly in order to minimize flight delay is the month May , in the US this is right before Summer begins in June . We can see that the highest delay is faced in June.



Barplot: Python



Barpot: R

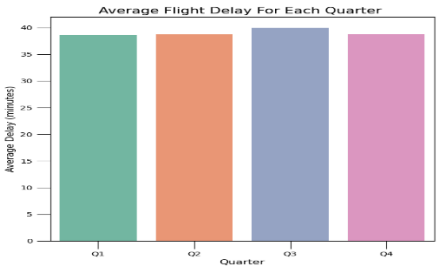
- In order to identify the best quarter of the year to fly we have to first identify the months of the year in to quarter's and then group the dataset by quarter and obtain the mean arrival and departure delay for each quarter and then we obtain a new average of the two delays and use this for our analysis. The bar plot and tables provided below make it easier to draw a conclusion, we see both table values in obtained in R and python below as well,

Quarter Of The Year	Total Average Delay
Q1	38.623465
Q2	38.756278
Q3	39.990759
Q4	38.819891

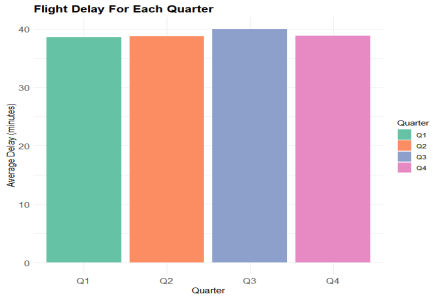
Table: Quarterly Delay (Python)

```
Quarter TotalAverageDelay
     Q1           38.62347
     Q2           38.75628
     Q3           39.99076
     Q4           38.81989
```

Table: Quarterly Delay (R)



Bar plot: Python



Bar plot: R

- According to the table given and bar plot shown the average delay faced in Q1 is less than the other quarters of the year, however the difference in delays faced in Q1 & Q2 is marginal and therefore making Q1 and Q2 the easiest time of the year to fly in order to minimize delays.

DO OLDER PLANES SUFFER MORE DELAYS?

- The analysis to be carried out requires the age of the plane to be calculated and in order to do so the cleaned dataset was merge with the dataset with plane-data, columns with similar data were renamed and the datasets were merged on 'TailNum' and unnecessary columns were dropped. The columns 'ArrDelay', 'DepDelay' and 'Manufactured_year' were grouped by 'Manufactured_year' and we obtain the mean arrival and departure delay for each manufacturing year and create a new column with the average of the two mean delays ,also any element in the column 'Manufactured_year' that was erroneous was dropped. An assumption was made regarding identifying planes as old and new, planes manufactured previous to 1982 were considered old and planes manufactured in 1982 or later were considered new. We then grouped the data by the plane type(old or new) and obtain the average delay for each type,

Plane_type	Total Average Delay
New	38.865538
Old	42.572099

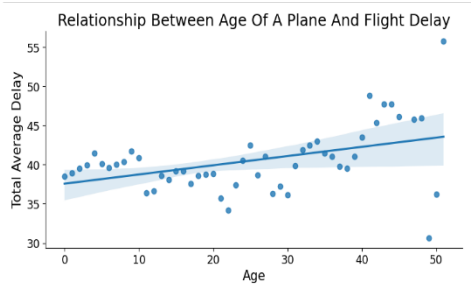
Table: Python

```
Plane_type TotalAverageDelay
<chr>                 <dbl>
New                    38.9
Old                    42.6
```

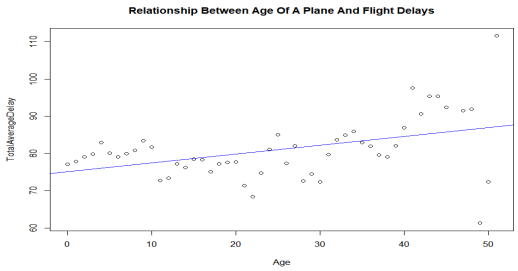
Table: R

- Older planes show a slightly higher delay than new planes, but it is important to remember these calculation have done with regard to the assumption made . The assumption was made with reference to https://www.paramountbusinessjets.com/faq/age-of-aircraft-safety-factor .

- The age of the plane was calculated by taking into account the year of manufacture and the year of the flight given in the dataset (2006 or 2007) , the new average delay calculated using the mean arrival and departure delay was then plotted in a scatterplot with the age of the plane to investigate if any relationship does exist between the two variables,



Scatter plot: Python



Scatter Plot: R

- The scatterplot shows there is some linear relationship between the age of the plane as we can see the trend line shows as age increases so does the average delay. The correlation between the two variables helps us to identify the strength of the linear relationship between the two variables,

```
df_new['Age'].corr(df_new['Total Average Delay'])
```

```
0.42599069311325305
```

Correlation: Python

```
cor(agg$Age,agg$TotalAverageDelay)
] 0.4259907
```

Correlation: R

we do see a positive correlation, which indicates a relationship does exist between the two variables.

HOW DOES THE NUMBER OF PEOPLE FLYING BETWEEN DIFFERENT LOCATIONS CHANGE OVER TIME?

- We try to identify any trends in the flying patterns of passengers over the years 2006 and 2007 , but since we do not have any information regarding passengers we use flight count between each origin to destination as a proxy assuming that the flight count and number of passengers are proportional. Both 2006 and 2007 dataset were was first filtered so that we remain with 'Origin', 'Dest' and 'Year' columns , after that both datasets were grouped by 'Origin' and 'Dest' columns and then we obtain a count of the number of rows in each group which is equivalent to the flight count from each origin to destination and finally the year column is renamed to the corresponding year of the dataset. Both dataframes were then merged on origin and destination column and we create a new column with the total flight count of 2006 and 2007 from each origin to destination and then we sort the dataset by total flight count , this will give us the top ten origin to destination which we identify as routes.

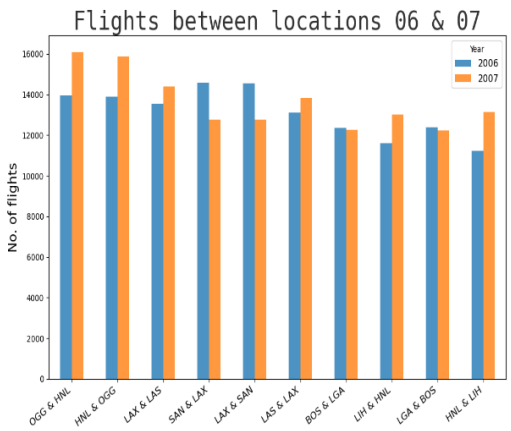
- We analyze the change in flight count between the top ten routes taken by passengers,

Origin	Dest	2006	2007	Total Flight Count	Route
OGG	HNL	13956	16099.0	30055.0	OGG & HNL
HNL	OGG	13898	15876.0	29774.0	HNL & OGG
LAX	LAS	13547	14385.0	27932.0	LAX & LAS
SAN	LAX	14594	12779.0	27373.0	SAN & LAX
LAX	SAN	14554	12767.0	27321.0	LAX & SAN
LAS	LAX	13122	13815.0	26937.0	LAS & LAX
BOS	LGA	12370	12263.0	24633.0	BOS & LGA
LIH	HNL	11602	13030.0	24632.0	LIH & HNL
LGA	BOS	12398	12222.0	24620.0	LGA & BOS
HNL	LIH	11235	13156.0	24391.0	HNL & LIH

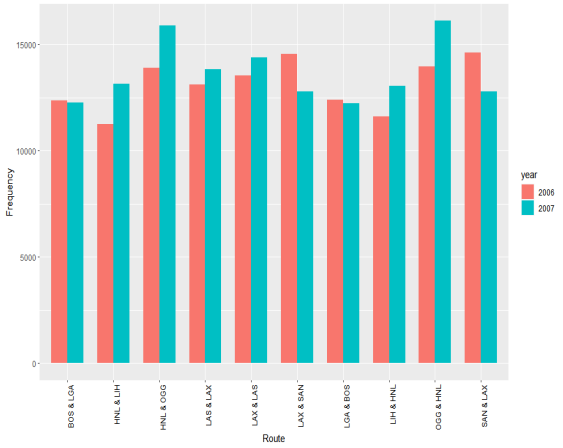
Table: Route Flight Count (python)

Origin	Dest	year2006	year2007	Total_Flight_Count	Route
OGG	HNL	13956	16099	30055	OGG & HNL
HNL	OGG	13898	15876	29774	HNL & OGG
LAX	LAS	13547	14385	27932	LAX & LAS
SAN	LAX	14594	12779	27373	SAN & LAX
LAX	SAN	14554	12767	27321	LAX & SAN
LAS	LAX	13122	13815	26937	LAS & LAX
BOS	LGA	12370	12263	24633	BOS & LGA
LIH	HNL	11602	13030	24632	LIH & HNL
LGA	BOS	12398	12222	24620	LGA & BOS
HNL	LIH	11235	13156	24391	HNL & LIH

Table: Route Flight Count (R)



Bar plot: Python



Bar plot: R

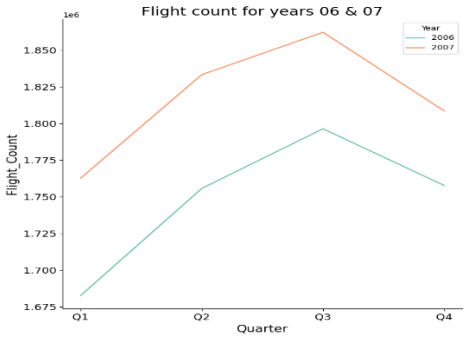
- In the year 2006 SAN-LAX was the most in demand route but in 2007 OGG-HNL had the highest flight count but in 2006 it was the 3rd most in demand route , we see 15.4 % increase in flight count in just a year and this could be due to many factors such as a rapid development in infrastructure or even a theme park etc..
  - We also analyze the total flight count over the two years with regard to the quarter of the year. We will be looking at the total flight count across all states in US for 2006 and 2007,

		Flight_Count
Year	Quarter	
2006	Q1	1682684
	Q2	1755692
	Q3	1796350
	Q4	1757689
2007	Q1	1762663
	Q2	1833264
	Q3	1862131
	Q4	1808530

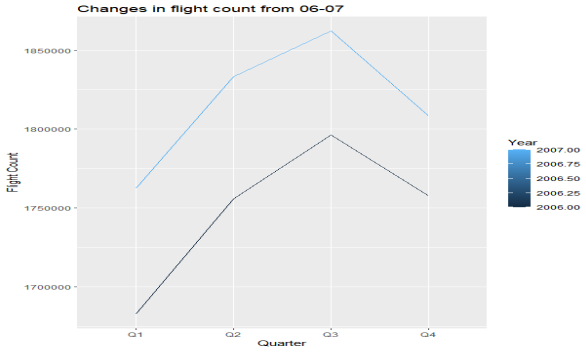
Table: Quarterly Flight Count (Python)

Quarter	Year	Flight_Count
Q1	2006	1682684
Q2	2006	1755692
Q3	2006	1796350
Q4	2006	1757689
Q1	2007	1762663
Q2	2007	1833264
Q3	2007	1862131
Q4	2007	1808530

Table: Quarterly Flight Count(R)



Line plot: Python

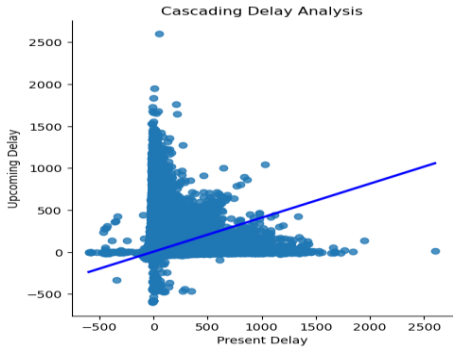


Line plot: R

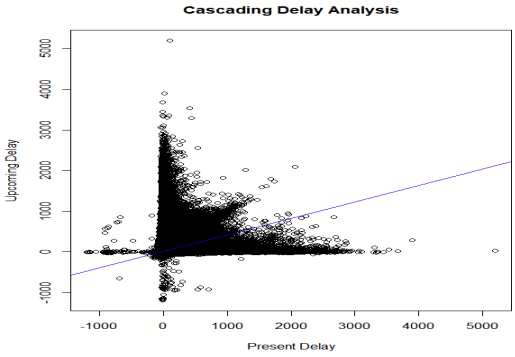
- The overall flight count in 2007 is greater than 2006 by 39% which is significant , we also see across both years flight count is lowest in Q1 which could be due to the holiday season . We also see flight count is highest across both years in Q3 which could be due to the beginning of summer in the US as summer starts at the end of Q2 and beginning of Q3.

CAN YOU DETECT CASCADING DELAYS IN ONE AIRPORT CREATE DELAYS IN OTHERS?

- We analyze if a departure delay caused at the origin airport causes an arrival delay at the connecting airport which in turn causes an arrival delay at the destination airport , the connecting airport may also be the destination airport depending on the route of the flight. We use the cleaned dataset and we create a new column called 'datetime' where we capture the year, month, day of month and 'DepTime' of the flight into a datetime format. The dataset is then grouped by 'TailNum' and 'datetime' , by doing this we are able to compare delay times between different planes in a chronological manner. By grouping we are also able to calculate mean 'ArrDelay' and 'DepDelay' and and create a new column with the average of the mean delays .The delay faced at the current airport was compared with the delay faced at its destination airport . A scatterplot was plotted to investigate the relationship between current delay faced and delay to be faced at the destination airport,



Scatter plot: Python



Scatter plot: R

- The scatterplot does show a positive relationship between delay faced at current airport and delay to be faced at the destination ,in fact we see a correlation of 0.405.

  ('Average Delay' is considered as delayed faced at present airport)

```
df['Average Delay'].corr(df['Upcoming Delay'])
```

```
0.40502728910321334
```

Correlation: Python

```
cor(df1$Average_Delay,df1$upcoming_delay)
] 0.4050273
```

Correlation: R

- In order to analyze this relationship further and make conclusions a cross tabulation was carried out and a table of observed values was obtained where 1 indicated a delay and 0 says otherwise , the table from python and R is given below,

upcoming delay	0	1
current delay		
0	5221665	2391893
1	2391893	4263298

```
              upcoming delay
current delay       0       1
            0 5221663 2391894
            1 2391894 4263298
```

- The table of observed values can be used to calculate the proportion of flights that had a delay at its present airport and the next destination airport , calculations show that 64.1% of flights that had a delay at its present airport had a delay at its next destination airport (4263298/4263298+2391893) and 31.4% of flights that faced a delay at its present airport would be able to reach its destination airport with no delay.

- We can further use the table of observed values to carry out a Chi – Squared hypothesis test to identify if there is indeed a rippling effect caused by flight delay ,
  Hypothesis,
  H0: There is no association between facing current delay and facing upcoming delay.
  H1: There is an association between facing current delay and facing upcoming delay.

  Test results from python(left) and R,

```
(1520474.5881165017,
 0.0,
 1,
 array([[4062463.03858621, 3551094.96141379],
        [3551094.96141379, 3104096.03858621]]))
```

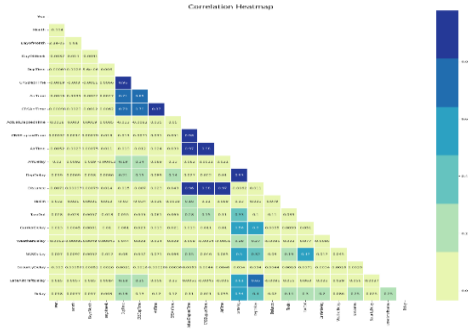
```
        Pearson's Chi-squared test

data:  table
X-squared = 1520473, df = 1, p-value < 2.2e-16
```

  Test statistic = 1 520 475 (python)
  P value = 0
  Degrees of freedom =1

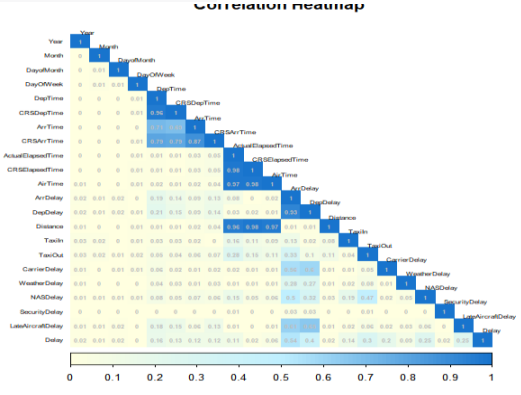
  - Therefore we reject the null hypothesis at 5% and 1% significance levels , we can conclude there is strong evidence to claim there is an association between facing current delay and facing upcoming delay. The ripple effect of flight delay does exist between airports.

USE THE AVAILABLE VARIABLES TO CONSTRUCT A MODEL THAT PREDICTS DELAYS

- A logistic regression model was built to predict flight delays and help passengers and airlines with their decision making with regard to air routes and travel destination, A logistic regression model is a statistical method that can be used to predict the probability of a binary outcome, such as whether a flight will be delayed or not. The 'ArrDelay' was used as the target variable and a binary column 'Delay' was created with two outcomes '1' and '0', where 1 denotes 'ArrDelay'>0 and 0 denotes 'ArrDelay'<0.(we choose 'ArrDelay' as our target variable as it had 0.93 correlation with 'DepDelay' therefore we choose 'ArrDelay' as our target variable to predict flight delay overall). We plot the correlation of variables in our dataset in a correlation plot (note that some categorical variables were dropped because they were not numeric entries and had over 100 categories and because they were erroneous e.g.: 'Origin')



Correlation Heatmap: Python

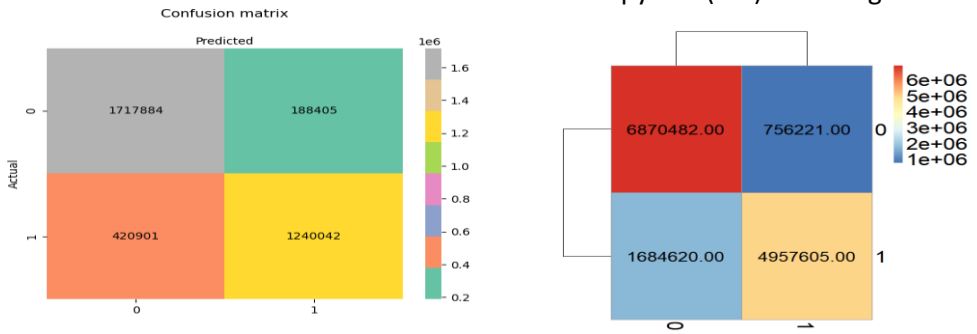


Correlation Heatmap: R

The higher the correlation the darker the color gets and lower the correlation the more yellow the plot is,

- The feature variables for the model were chosen with the use of the heatmap , variables such as 'CRSDeptime' , 'DepDelay' , 'DayOfWeek' , 'Month' ,'TaxiIn' , 'TaxiOut' and 'LateAircraftDelay' . 'ArrDelay' is calculated at the destination airport therefore information regarding 'CRSDeptime' (scheduled departure time) ,'DepDelay' and 'LateAircraftDelay' will be available and can be used as feature variables in the model. Variables with some correlation with 'ArrDelay' such as 'Month' and 'DayOfWeek' was also included in the model.Variables such as 'WeatherDelay' , 'NASDelay' , 'SecurityDelay' etc.. were not included in the model as this data will not be available until the plane is at its destination airport.

  Constructing the model
- The dataset is split into training and testing data which contain only our feature variables and target variable , the split we take 75/25 where 75% is training data and 25% is testing data. After the data is split we scale all variables except the target variable in both the train and test data , it is important to scale our data as our variables have different units of measure . In our model we have applied a standard scaling approach.
- The model is fit with the training and testing data after the scaling process and then we obtain a confusion matrix . The confusion matrix from python(left) and R is given below,



- The confusion matrix obtained was used to evaluate the performance of the model with regard to predicting delays,

- Performance metrics of the model made in R.

accuracy	precision_delay	precision_nodelay	recall_delay	recall_nodelay	f1_delay	f1_nodelay
0.8289	0.7463772	0.9008456	0.8676507	0.8030859	0.8024578	0.8491614

- Performance metrics of the model made in python.

```
              precision    recall  f1-score

    No Delay       0.80      0.90      0.85
       Delay       0.87      0.75      0.80

    accuracy                           0.83
   macro avg       0.84      0.82      0.83
weighted avg       0.83      0.83      0.83
```

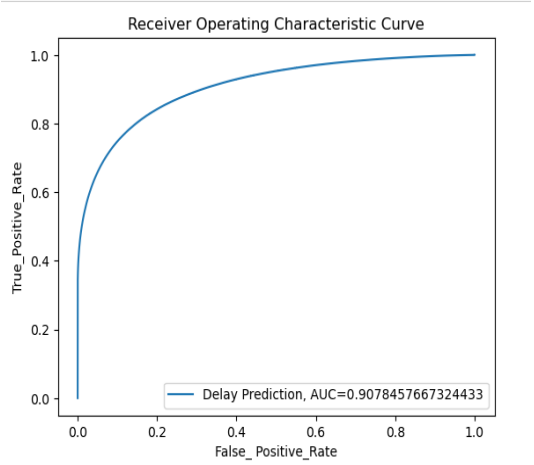
The reason we see different scores from the same model when done in R and python is due to the random state factor when it comes to selecting a sample for our train and test data, differences in the implementations of the algorithms used in R and Python can also lead to variations in the model performance. Let us now interpret the performance metrics,

1. Accuracy shows the models ability to accurately make the prediction if a delay will be faced , we see an accuracy of 83% in both models which is satisfactory but it important to note that accuracy scores are affected by class imbalances in the dataset .
2. Precision of the model tells us the exact number of flights that may face a delay or no delay, we see a different scores for precision from the model constructed in R and python but overall both models give satisfactory scores. E.g.: The model in python gives a precision score of 0.87 for no delay which means that out of all the flights that the model predicted to be not delayed, 87% of them were actually not delayed.
3. Recall score tells us how many of the actual delay or no delay flights are correctly identified by the model. E.g.: The model in R gives a recall score of 0.80 for no delay means that out of all the flights that were actually not delayed, the model correctly predicted 80% of them as not delayed.
4. F1 score is the harmonic mean of the precision score and recall score. It is a single score that combines the precision and recall into one performance metric. E.g.: The model in python gives a F1 score of 0.85 for no delay which means that the model has a good balance between precision and recall when the model predicts no delay.

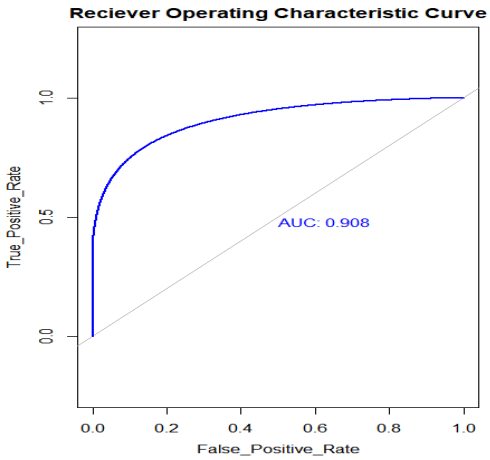
- Overall we the performance metric of both the models made in R and Python are satisfactory with regard to its ability to predict flight delay.

Receiver Operating Characteristic Curve

➢ The Receiver Operating Curve (ROC) is a graphical representation of the performance of our model which is a logistic regression model, ROC curve shows us the relationship between the true positive rate and false positive rate as the classification threshold of the model varies when predicting a flight delay, The true positive rate is the proportion of flights actual delayed that are correctly identified as delayed by the model and the false positive rate (FPR) is the proportion of actual non-delayed flights that are incorrectly identified as delayed by the model.



Plot: Python



Plot: R

- We see both ROC curves show an AUC score (area under the curve) of 0.91 approximately which shows high overall performance of the model in identifying delayed flights while minimizing false positives.